(1)

RL-TR-94-95
Final Technical Report
July 1994

AD-A285 534

FLEXIBLE COORDINATION IN RESOURCE-CONSTRAINED DOMAINS

Carnegie Mellon University

Sponsored by
Advanced Research Projects Agency
ARPA Order No. 7407

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

Rome Laboratory
Air Force Materiel Command
Griffiss Air Force Base, New York

RL-TR-94-95 has been reviewed and is approved for publication.

APPROVED: Albert G Frantz

ALBERT G. FRANTZ
Project Engineer

FOR THE COMMANDER: [illegible]

JOHN A. GRANIERO
Chief Scientist
Command, Control & Communications Directorate

FLEXIBLE COORDINATION IN RESOURCE - CONSTRAINED DOMAINS

Dr. Stephen F. Smith
Dr. Katia P. Sycara

Contractor: Carnegie Mellon University
Contract Number: F30602-90-C-0119
Effective Date of Contract: 28 September 1990
Contract Expiration Date: 28 September 1993
Short Title of Work: Flexible Coordination in Resource-Constrained Domains
Program Code Number: 3E20

Period of Work Covered: Sep 90 - Dec 93

Principal Investigator: Dr. Stephen Smith
Phone: (412) 268-8811

RL Project Engineer: Albert G. Frantz
Phone: (315) 330-4031

This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by Albert G. Frantz, RL (C3CA), 525 Brooks Road, Griffiss AFB NY 13441-4505 under Contract F30602-90-C-0119.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE July 1994	3. REPORT TYPE AND DATES COVERED Final Sep 90 - Dec 93

4. TITLE AND SUBTITLE FLEXIBLE COORDINATION IN RESOURCE - CONSTRAINED DOMAINS	5. FUNDING NUMBERS C - F30602-90-C-0119 PE - 62301E PR - 7407 TA - 00 WU - 01
6. AUTHOR(S) Dr. Stephen F. Smith and Dr. Katia P. Sycara	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Carnegie Mellon University School of Computer Science Pittsburgh PA 15213	8. PERFORMING ORGANIZATION REPORT NUMBER N/A
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Advanced Research Projects Agency 3701 North Fairfax Drive Arlington VA 22203-1714 Rome Laboratory (C3CA) 525 Brooks Road Griffiss AFB NY 13441-4505	10. SPONSORING/MONITORING AGENCY REPORT NUMBER RL-TR-94-95

11. SUPPLEMENTARY NOTES

Rome Laboratory Project Engineer: Albert G. Frantz/C3CA/(315) 330-4031

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The broad goal of this research, which was carried out as part of the ARPA/Rome Laboratory Planning Initiative, has been to investigate the use of constraint-based scheduling frameworks and techniques as a basis for more accurate and more flexible decision support as various stages of the military crisis-action planning, deployment and employment process. These investigations have led to the development of DITOPS, a constraint-based, transportation scheduling prototype with facilities to support decomposition and distributed decision-making.

14. SUBJECT TERMS Scheduling, Constraint-based, Distributed, Artificial Intelligence	15. NUMBER OF PAGES 122
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL

1 Introduction and Overview

In this final report, we summarize the research performed under Advanced Research Projects Agency (ARPA) contract F30602-90-C-0119, Flexible Coordination in Resource-Constrained Multi-Agent Domains. The broad goal of this research, which was carried out as part of ARPA/Rome Laboratories Planning Initiative (PI), has been to investigate the use of constraint-based scheduling frameworks and techniques as a basis for more accurate and more flexible decision support at various stages of the military crisis-action planning, deployment and employment process.

The scheduling problems faced in the crisis-action planning domain present significant technical challenges. The problems are large-scale, the planning and execution environment is dynamically changing, and solutions must integrate the actions of multiple decision-makers. Coordinated decision-making about resource apportionment and allocation is required at different levels of detail over different time scales, including (1) the ability to assess deployment transportation feasibility at early stages in the mission planning process and determine overall transportation asset requirements, (2) the ability to generate increasingly detailed deployment schedules that satisfy operational constraints, make efficient use of transportation assets and balance conflicting mission objectives, and (3) the ability to rapidly adapt schedules in light of changing (or evolving) circumstances. Existing transportation planning/scheduling systems and mobility analysis models generally operate with models that simplify important domain constraints (limiting the confidence that can be placed in results), inadequately manage problem complexity (sacrificing optimality), provide only limited support for many planning tasks (forcing extensive manual analysis and manipulation of system inputs and outputs), are inflexible in reactive decision making contexts (limiting responsiveness to change), and provide no support for coordination of different planning tasks (increasing overall planning time).

Toward the development and demonstration of constraint-based scheduling technologies that overcome these limitations, our research objectives have been two-fold:

1. to extend and generalize previously developed theories and techniques for constraint-directed scheduling for application to the military crisis-action logistics domain, and to evaluate their effectiveness in various decision-support contexts, and

2. to augment this constraint-directed scheduling methodology with protocols and coordination strategies to support integrated decision-making by multiple transportation scheduling agents.

We have taken a specific constraint-based scheduling technology, the OPIS manufacturing scheduling system, as our starting point and have focused on generalization and application of its multi-perspective scheduling technology to the deployment scheduling problem. These investigations have led to the development of DITOPS, a constraint-based, transportation scheduling prototype with facilities to support problem decomposition and distributed decision-making (DITOPS stands for Distributed Transportation Scheduling in OPIS). In parallel, we have also conducted more basic research into new constraint-based scheduling procedures and into mechanisms for coordinating the scheduling actions of multiple agents.

Methodologically, our goal has been to demonstrate the feasibility and potential of constraint-based scheduling technologies under realistic crisis-action planning assumptions. Our augmentation and generalization of the capabilities defined in the underlying OPIS scheduler have been focused by detailed analysis of the military crisis-action planning process, the characteristics and scheduling requirements in this domain, and the capabilities of current transportation scheduling tools. We have experimentally evaluated the resulting DITOPS system on large scale transportation scheduling problems defined within the ARPA PI Common Prototyping Environment (CPE), and additional functional capabilities have been demonstrated through the conduct of Technology Integration Experiments (TIEs) with planning technologies developed at both BBN (FMERG) and SRI (SOCAP). We have also exported scheduling support capabilities provided by DITOPS for use within the TARGET Integrated Feasibility Demonstration system (IFD 3).

1.1 The DITOPS Transportation Scheduler

The constraint-based scheduling concepts and techniques implemented and demonstrated in the DITOPS scheduler offer several sources of leverage in addressing crisis-action deployment problems:

- incorporation of additional deployment constraints - One basic pre-requisite for development of realistic deployment schedules is an ability to accurately model and account for all important domain and problem constraints. DITOPS is able to generate transportation schedules that take into account important classes of constraints that are not currently modeled in current transportation planning practice (e.g., the temporal precedence and synchronization constraints between "TPFDD" movement records). It is also capable of enforcing other classes of constraints that are modeled in practice but are typically ignored by other transportation scheduling tools (e.g., enforcement of "earliest arrival date" (EAD) constraints).

- higher quality schedules - Through reliance on techniques that use information about constraint interactions to guide the scheduling process, DITOPS is able to produce schedules that better optimize transportation objectives than schedulers that operate with conventional, simulation-based procedures. Dramatic improvements to deployment "closure profiles" have been demonstrated, for example, in comparative analysis experiments with other representative TPFDD scheduling approaches at comparable computational cost.

- incrementality and reactive capabilities - The constraint-based scheduling procedures utilized to generate schedules in DITOPS are incremental in nature and thus equally applicable to the problem of incrementally revising a schedule in response to change. DITOPS provides a variety of rescheduling methods, each designed to locally revise specific decisions in the schedule while emphasizing specific reoptimization objectives. Through use of constraint propagation and analysis methods, DITOPS provides guidance as to what decisions in the schedule must be revised in a given reactive context, what opportunities exist for non-disruptive change, and what reoptimization objectives should be emphasized, all of which can be used to direct the schedule revision process.

- Flexibility to support different planning tasks - DITOPS provides scheduling procedures that can be flexibly adapted to obtain different functional capabilities. One level of flexibility is provided by the ability to selectively specify "relaxable" constraints and overlay preference structures (in the form of a utility function) on their satisfaction. This enables the scheduler to be configured to address qualitatively different deployment questions. For example, arrival dates can be specified as relaxable to perform closure analyses under specific asset apportionment assumptions. (This is the specific task for which most current transportation analysis tools are designed.) It is also possible to instead designate asset capacity constraints as relaxable to estimate the resources required to achieve mission closure dates. (This is a task that is typically not addressable with current tools.)

  Another level of flexibility stems from the extensibility of the underlying modeling framework and scheduling infra-structure. Exploiting object-based representation techniques and an object-oriented programming methodology, the DITOPS scheduler is explicitly designed for extension/customization of modeling capabilities (to effectively incorporate the important idiosynchracies of a given planning domain) and reuse of component constraint management and scheduling techniques (to rapidly adapt scheduling support capabilities to fit the requirements of different decision-support tasks and applications).

1.2 The Larger Vision

From a broader perspective, the constraint-based scheduling capabilities demonstrated in the DITOPS scheduler point the way toward a different paradigm for decision support than is provided in current transportation and mobility analysis tools; a paradigm that more directly matches the requirements and characteristics of the transportation planning and scheduling process. Construction of transportation schedules in practice is an *iterative reactive process.* An initial schedule is built, problematic or unsatisfactory aspects of the result are identified, requirements are relaxed or strengthened (typically through negotiation with other planning agents), schedule modifications are made and so on. Throughout this process, the current schedule provides the planner(s) with an important nominal reference for identifying, specifying and communicating changes, and there is considerable pragmatic value in an ability to retain continuity (or localize change) in the solutions that are produced across iterations. Such an ability allows the planner to impose structure on an otherwise overwhelmingly complex search process and to converge in a more focused fashion to an acceptable overall solution. Likewise, as unexpected events occur in the execution environment (e.g., changes to mission requirements, unexpected unavailability of lift capacity), it is important to preserve continuity in domain activity while making those schedule changes necessary to restore feasibility and insure continued attendance to overall mission performance objectives. Both of these aspects of the scheduling process place a premium on incremental, reactive scheduling capabilities.

In contrast to these decision support requirements, current transportation scheduling tools are typically *batch-oriented* solution generators. In commonly used simulation-based technologies, for example, problem input parameters and constraints are specified, the system is run to produce a schedule, and the result is examined for acceptability. In reacting to either unsatisfactory properties of the generated schedule (e.g., unacceptable late closures) or changing circumstances in the world (e.g., the unexpected loss of port capacity), the human planner is forced to hypothesize how changes to system inputs might affect the solution that is produced, and has no control over what aspects of the solution actually will change when the system is rerun with specified input parameter changes. Consequently, there can be considerable "thrashing" in the solutions generated from run to run, and it is quite cumbersome to enforce commitment to specific aspects of any given solution.

Constraint-based scheduling procedures, alternatively, by virtue of their inherently incremental and decomposable nature, enable an *interactive* decision-support paradigm based directly on focused, incremental change to the current solution. Constraint-based scheduling procedures manipulate schedules "from the side" (i.e., placing and rearranging activities on a time line in accordance with resource and process constraints, as opposed to simulating execution forward in time and recording the activity time and resource assignments that result as a by-product), providing a natural framework for selective user exploration and comparison of alternative assumptions, and for direct, controlled convergence to an acceptable solution. It is possible to incrementally commit to subsets of decisions in the current solution (e.g., "locking down" decisions associated with particular forces or transportation resources), to likewise designate sets of activities or regions of the time line that require change/improvement, and to specify constraint changes (e.g., addition of lift capacity, routing changes) to be taken into account as the schedule is revised. The (reactive) scheduling methods implemented in the DITOPS scheduler provide the types of functionality required to support this interactive decision-making model.

Our vision of transportation scheduling tools of the future are decision-support environments similar in spirit to current day spreadsheet programs; sets of scheduling decisions and solution constraints are interactively manipulated by the user at levels consistent with user-task models, with the system applying appropriate (re)scheduling procedures to implement user actions (i.e., manage the details) and provide immediate, localized consequences of each change. Constraint analysis techniques will contribute additional leverage to this incremental scheduling process, providing guidance to users in identifying the principal causes of observed solution deficiencies (e.g., resource bottlenecks) and in analyzing various decision-making options.

The decision-support capabilities we envision are illustrated by the the following interactive "TPFDD" generation scenario:

1. **Evaluate initial schedule**

   Starting with a set of deployment requirements and initial estimates as to apportioned transportation resources, a USTRANSCOM planner invokes the system to generate an initial schedule that satisfies stated resource capacity and utilization constraints and minimizes late closures. Upon inspection of the results too many late closures are discovered.

2. **Identify principal bottleneck**

   System analysis of the constraints contributing to these results indicates the principal source of lateness to be insufficient throughput capacity at the designated final port of debarkation, POD1.

3. **Propose a solution**

   The planner responds to this information by introducing a second port of debarkation, POD2, into the scenario and indicating that POD1 arrivals be rescheduled to exploit the additional capacity provided by POD2. The number of late closures is substantially reduced by this action.

4. **Identify next bottleneck**

   Analysis of the resulting schedule now indicates that the remaining late closures stem from inadequate sea lift capacity during week 2 of the deployment.

5. **Engage in clarification dialog**

   Several "what-if" actions are carried out to determine additional resource requirements and to clarify alternative options for eliminating late closures:

   (a) Late movements are rescheduled with the specification that lift capacity constraints may be relaxed (i.e., additional assets may be added), which indicates that two additional transports are needed to meet all specified arrival dates.

   (b) The sea mode assignment associated with the remaining late arrivals is eliminated to determine whether excess air lift capacity can be utilized to resolve the problem. Results of this action indicate that only 50% of the late cargo can be accommodated by available air capacity (due in part to capacity limitations and in part to the cargo carrying restrictions of available aircraft types).

6. **Locate additional resources**

   At this point, the user decides that acquisition of additional sea assets is the best option and proceeds to obtain use of two commercial transports during the 2nd week of the mission.

7. **Propose a solution**

   The additional lift capacity is added to the model and late movements are rescheduled to complete by their requested arrival dates.

Given the complexity and scale of transportation scheduling problems, a crucial component of such an environment is a framework for interaction that enables the user to visualize, analyze and manipulate solutions at multiple, aggregate levels. The current DITOPS user interface has taken some initial steps in this direction, providing facilities for graphically visualizing and manipulating resource schedules and capacity constraints at different levels of aggregation. But significant challenges remain in effectively bridging the gap between user and system models of transportation schedules; this constitutes a major focus of our current research.

1.3 Organization of the Report

In Section 2, we summarize the major accomplishments and contributions of the research effort. In Section 3, we summarize the concepts and techniques underlying the current DITOPS transportation scheduler, the experimental results obtained in the context of TPFDD scheduling, the interactive, reactive capabilities for what-if exploration and incremental schedule revision that have been demonstrated, and the extended distributed scheduling prototype implemented. In Section 4, we summarize the additional capabilities and results obtained through participation in various Technology Integration Experiments with other PI sponsored organizations. Finally in Sections 5, 6 and 7, we provide summaries of our supporting basic research in constraint posting scheduling, distributed constraint satisfaction and multi-specialist problem solving respectively.

2 Summary of Accomplishments

Our research has produced the following major accomplishments:

- *Ontological primitives for modeling transportation scheduling problems* - We have developed a general scheduling ontology (in the form of an exportable class library) that enables specification of transportation domain models that incorporate all important constraints in any given transportation scheduling context. The ontology is explicitly designed to support:

  - *Realistic models of resource allocation constraints, objectives and preferences* - The ontology provides primitives for differentially modeling a wide range of resource types (resuable, consumable, shared, atomic, composite, mobile, stationary, etc.) and allocation constraints (capacity limits, cargo compatibility restrictions and preferences, mobility and availability constraints, allocation preferences, etc.). Likewise, primitives are defined for modeling the component activities of transportation plans (e.g., transporting, loading, unloading, processing, etc.), the temporal relationships that exist among them (e.g., multi-leg plans, synchronized air/sea movements, etc.), absolute timing constraints and preferences on their execution, and their resource requirements.
  - *Multi-level models* - The ontology provides structures and protocols for constructing hierarchical descriptions of transportation processes and required resources, allowing the level of detail at which allocation decisions are considered to be selectively and dynamically varied according to planning context (e.g., high-level course of action analysis, tpfdd-level feasibility analysis, detailed port scheduling) and domain characteristics (e.g., the criticality of various constraints).
  - *Extensibility and reuse* - The ontology provides general protocols for combination and extension/customization of concepts to capture the important idiosynchracies of a given transportation scheduling application.

- *Constraint-based techniques for transportation scheduling* - We have extended and adapted the multi-perspective scheduling techniques of OPIS to incorporate the dominant characteristics of transportation scheduling problems and enable multi-perspective construction and revision of transportation schedules. At the infra-structure level, we have developed constraint management techniques to enforce cargo "batching" constraints, to enable "splitting" of move requirements too large to be accommodated by a given asset, to incorporate capacity requirements that involve multiple resources (e.g., lift asset and port capacity), and to account for resource location constraints. We have developed two general scheduling (or rescheduling) procedures that localize decision-making along two distinct foci: A "resource scheduler", which constructs (or revises) some portion of the schedule of a designate transportation asset (or set of assets), and a "movement scheduler", which constructs (or revises) the schedule of a designated set of temporally connected move requirements (e.g., a multi-leg trip).

More specialized reactive methods for shifting movement schedules and redirecting movements to nearby destinations have also been developed to provide capabilities for qualitatively different types of reactive change.

- *Demonstration and analysis of capabilities in the domain of "TPFDD" scheduling* - In collaboration with BBN (Cambridge), a comparative analysis of TPFDD level deployment scheduling capabilities was carried out with respect to PFE. On the "MEDCOM problem scenario" that was utilized in IFD2, the DITOPS scheduler was shown to (1) to produce deployment schedules for various sea and air assets with 25-50% reduction in movement tardiness over PFE (assuming comparable constraints), and (2) provide an ability to enforce important constraints (e.g., earliest arrival dates) that are currently not handled within PFE. In this latter case, a 6% reduction in movement tardiness over the PFE schedule was still obtained for sea cargo movements (i.e., even when additional, more restrictive constraints were enforced). Another TIE experiment with BBN demonstrated the use of DITOPS scheduling capabilities in support of decisions earlier in the TPFDD generation process, specifically the ability to make transport mode assignments that take better advantage of the capabilities of apportioned transportation assets through generation of aggregate level deployment schedules. Capabilities for incrementally revising deployment schedules to account for changes in problem constraints (e.g., the unexpected unavailability of a POE, a reduction in lift capacity) have also been developed and demonstrated, supporting both reactive management of deployment schedules as well as a basis for pro-actively evaluating the impact of various possible scenarios.

- *Re-engineering and porting of OPIS/DITOPS modeling and scheduling infrastructure into a PI-compatible software/hardware environment* - As part of this project, we have ported the underlying OPIS scheduler from a TI Explorer environment including KnowledgeCraft to CommonLisp/CLOS/CLIM on a Sun Workstation. The new software architecture is heavily based on object-oriented representation and programming techniques, and is organized to promote rapid adaptation and configuration of component scheduling functionality (or tools) into new scheduling services that fit the requirements of specific client applications. The modeling and scheduling infrastructure is defined according to a layered system semantics (which is implemented in the form of class libraries). At the base of the system is an extended object system which adds necessary "frame-like" representation capabilities. Using these basic capabilities, basic kernel scheduling components are then defined (e.g., constraint propagation techniques, general purpose modeling primitives, capacity analysis techniques). More specialized system components (e.g., the transportation scheduling methods and heuristics of DITOPS) are in turn defined by composing relevant kernel scheduling services, and finally, those capabilities specific to a particular application domain are configured (in our work thus far, relating primarily to the joint strategic deployment scheduling domain).

- *Interactive Transportation Scheduling* - We have integrated the ported infra-structure with graphical schedule visualization and manipulation capabilities to provide a flexible interactive environment for construction and management of transportation schedules. Utilizing the system's hierarchical domain model, the user interface promotes interaction at aggregate levels. The user can view resource schedules, presented graphically as usage profiles over time, at different levels of detail (e.g., for an individual ship, for the cargo ship fleet, for all transportation lift assets). Building in part on capabilities provided by the CPE SciGraph package, activity-centered views (e.g., movement closure profiles) can also be examined for a graphically selected portion of any resource schedule. Changes in availability of various resources (e.g., indicating port closures, addition or loss of transportation assets) can be graphically communicated, utilizing the reactive scheduler to examine effects. Users may also specify changes to various scheduling preferences and objectives utilized by the system (e.g., preferring use of large ships to small ships) to explore the consequences of various tradeoffs.

- *Integration of resource analysis capabilities into higher-level deployment planning processes* - Using the ported infra-structure, we have configured and exported an employment plan constraint checking/scheduling module for integration by BBN/San Diego into the TARGET IFD-3 system. Also, in collaboration with SRI, we developed and provided a resource capacity analysis capability to support plan evaluation within SRI's SOCAP course of action (COA) plan generator.

- *Development and demonstration of distributed, multi-level deployment scheduling* - Through analysis of current transportation planning practice, criteria for problem decomposition (scope, granularity, types of decisions) were identified, leading to the definition of a multi-level model and organizational structure for distributed transportation scheduling and control. We developed and implemented a communication and coordination infra-structure to support this distributed model, and demonstrated its use in integrating the scheduling activities of a global (e.g., "transcom" level) agent and multiple port schedulers.

- *Development and validation of new "constraint-posting" scheduling techniques* - We have developed new procedures for constructing schedules which, in contrast to conventional approaches to scheduling, operate with the more general representational assumptions of contemporary temporal planning frameworks, and thus provide natural opportunities for tighter integration of planning and scheduling processes. Experimental work thus far has concentrated on calibrating performance leverage with respect to classical scheduling approaches on published benchmark problems, and the results obtained thus far are quite impressive. We have demonstrated (1) an ability to produce solutions comparable to micro-opportunistic, "bottleneck-based" approaches on constraint satisfaction problems with orders of magnitude speedup, and (2) an ability to outperform the best known approximation algorithms developed in the Operation Research community in various schedule optimization contexts.

- *Development and analysis of frameworks for cooperative, multi-agent decision-making* - We have developed an approach for distributed constraint satisfaction based on (1) partitioning the set of constraints into subsets of different types and (2) associating responsibility for enforcing constraints of each type with different sets of specialized agents. Variable instantiation is the joint responsibility of different teams of these specialized agents, and the final solution emerges through incremental local revisions of an initial, possibly inconsistent, instantiation of all variables. Experimental evaluation of the approach on constraint satisfaction scheduling problems has shown this distributed approach to also perform comparably to micro-opportunistic, bottleneck-based procedures with much greater computational efficiency.

We have also developed a model for collaborative decision-making by teams of specialists, each with unique areas of expertise and limited understanding of the expertise of other agents. The approach is based on a partitioning of agent knowledge into expert and naive models. The naive portion of agents' models provides both a common language and the inferential skeleton needed for the development of shared models. Model refinement occurs when problem solving reaches an impass; structured communications among agents are tied to model manipulations, which dynamically alter agents' evaluations and justifications, and results in more precisely directed overall search.

- *Contributions to PI integration and infra-structure activities* - We have been active in supporting numerous joint activities of the PI: serving as co-chair of the working group responsible for developing the PI's overall vision or "Technical Roadmap" for planning and scheduling technology development and identifying critical experiments, and serving as co-chair of the Scheduling Technology Working Group. We have also made contributions to the knowledge representation working group (included in the Knowledge Representation Specification Language (KSRL) document) and to the development of the Common Plan Representation (CPR). We have supported BBN in its development of the Common Prototyping Environment (CPE), including insertion of the DITOPS scheduler into the CPE and development of the interface modules to make the system accessible as a knowledge server through the CRONUS inter-module communication infra-structure.

3 Technical Overview of DITOPS

DITOPS is an advanced tool for generation, analysis and revision of crisis-action logistics schedules. The system incorporates concepts of constraint-directed scheduling developed within the OPIS manufacturing scheduling system at CMU, together with extensions to address the specific characteristics of transportation scheduling problems. Using DITOPS, we have demonstrated an ability to efficiently generate higher quality schedules than conventionally used simulation approaches on large-scale deployment scheduling problems while simultaneously satisfying a wider range of deployment constraints. Just as important, DITOPS also provides flexible capabilities for incrementally revising schedules in response to changed constraints. These capabilities allow schedules to be reactively updated to reflect unexpected events that occur during schedule execution (e.g., the closing of a port due to bad weather) while preserving continuity in scheduled activities wherever feasible. They also allow for efficient, controlled convergence to acceptable (or improved) solutions during advanced planning; as adjustments to various scheduling constraints and preferences are made by human planners in response to observed solution deficiencies (e.g., too many late closures), DITOPS can provide immediate, localized feedback of the effects of these changes on the current schedule. DITOPS is implemented using object-oriented representation and programming techniques, providing an extensible modeling and scheduling framework that enables straightforward system customization to account for the principal constraints and objectives of different scheduling domains.

The DITOPS scheduling framework is founded on three basic principles:

1. decision-making must be rooted in a representational framework sufficient to [illegible]ture important domain constraints and scheduling preferences - DITOPS provides a general framework for modeling transportation processes, required resources, movement requirements and shipments, which can be instantiated in any specific problem domain to encode all relevant temporal synchronization and resource utilization constraints on solution feasibility. Specific types of constraints (e.g., deadlines) can be selectively modeled as relaxable preferences, and domain models are defined hierarchically to enable different levels of constraint specificity (e.g., to match their relative importance in a given problem context).

2. dynamic look-ahead analysis of the structure of problem constraints is the key to efficient and effective scheduling - At the core of DITOPS is an incremental, reactive framework for generating and revising schedules [Smith 93], which relies on repeated analyses of current problem constraints (e.g. projected resource contention, current scheduling conflicts) to focus attention toward most critical decisions and tradeoffs, and to select appropriate decision-making (or decision revision) procedures.

3. large-scale problem solving invariably involves multiple decision-makers and distributed decision-making - The DITOPS scheduler has been augmented with mechanisms for inter-agent coordination, and initial protocols and interaction strategies consistent with military transportation planning and control requirements have been implemented.

In the subsections below we first summarize the technical approach taken to representation and decision-making within the DITOPS transportation scheduler. We then summarize the performance and interactive/reactive capabilities of the core scheduler. This is followed by an overview of the extended prototype developed for distributed multi-level transportation scheduling. A summary of additional functionality that was configured using components of the scheduler to support various course of action planning processes is presented in Section 4.

3.1 Modeling Transportation Scheduling Constraints

The DITOPS scheduler operates with respect to a hierarchical model of the resources and resource allocation constraints of a given application domain. The use of a hierarchical model serves three basic purposes. First, it enables decision-making at different levels of abstraction to support different stages of the overall planning process (e.g. high level capacity analysis, determination of transport modes, detailed asset assignments and movement schedules). Second, it provides a basis for focusing the scheduler's search process when scheduling (or rescheduling) at a specific level of detail. Finally, it provides a structure for decomposing and distributing a transportation scheduling problem among multiple agents, and a basis for coordinating multi-agent decision-making across different levels.

A DITOPS model of a given application domain is composed from from an extensible set of pre-defined primitives, which provide object structures (i.e., a class library) for specifying various transportation scheduling constraints and associating an appropriate operational semantics. A transportation scheduling model is specified as a relational configuration of five basic types of "building blocks":

- *Resources* - Resource objects represent the various assets, equipment, and facilities required to carry out deployment requests. A variety of specialized resource classes are defined to support specification of different types of resources. These resource types include unit capacity resources, which must be allocated exclusively to a single request (e.g., a loading/unloading crane), batch capacity resources, which can simultaneously accommodate multiple requests over the same interval (e.g., a sea barge or tanker ship), and a variety of disjunctive and conjunctive aggregate capacity resources, where capacity can be simultaneously allocated to multiple requests without temporal synchronization (e.g., a C-5 plane fleet, a tanker ship fleet, a seaport). Atomic resources can be grouped through the definition of composite resources (e.g. individual tankers into a tanker fleet into an overall sea fleet; unloading equipment, storage capacity, parking places, etc, into a port) to provide consistent descriptions of resources and utilization constraints at multiple levels of abstraction. Such resource models provide the basis for hierarchical specification of transportation processes.

  A central component of each resource class specialization is a set of methods for managing and querying a representation of available capacity over time. These methods define the resource class's allocation semantics from the standpoint of scheduling and control decision-making. Resources can also be distinguished as mobile (a ship) or stationary (a port); the former case implying the representation (and management) of a second dynamically changing property, location. Other utilization constraints associated with resource descriptions and enforced by allocation methods include constraints on capabilities (e.g., subset of commodity types that can be moved by a given type of transport asset), resource capacity constraints, and batching constraints (e.g., incompatibilities among commodity types that might be carried simultaneously).

- *Operations* - Operation objects are used to represent the constituent actions of transportation processes (or plans). Generally speaking, an operation specifies the set of constraints and effects that define a particular activity (resource requirements, duration constraints, temporal relations relative to other activities, cargo involved). Like resources, a taxonomy of specializations are defined to characterize different activity types. For example "transport operations" specify an origin (POE) and destination (POD), which imposes a setup requirement that the allocated transportation asset be at the origin at the start of the operation and an effect that leaves the allocated asset at the destination location. "Load" and "unload" operations, alternatively, do not change asset location. Through association of temporal relationships and/or synchronization constraints to other operations, operation descriptions can be composed into larger transportation processes. Operations can also be organized hierarchically to provide descriptions of transportation processes at different levels of resource specificity.

- *Move Requirements* - Move requirement objects represent the input requests that the scheduler must attend to. These descriptions specify cargo characteristics (e.g., cargo and commodity types), quantities, origin and destination of the movement (e.g., POE and POD), and relevant absolute time constraints (e.g., ALD, EAD, LAD, etc.), as well as any temporal relations and/or synchronization constraints with other move requirements (e.g., that two movements must arrive within a day of each other). In the context of deployment scheduling, move requirements correspond directly to individual TPFDD records.

- *Shipments* - Shipment objects represent the actual cargo entities (or "packages") that are associated with individual transport operations (e.g., the 25th infantry division, 1000 CBarrels of POL, etc.). Shipments are created in response to the cargo specifications given in move requirements. Generally speaking, accomplishment of a given move requirement may necessitate the transport of several shipments (i.e., require multiple trips), since a move requirement's lift requirements may exceed the capacity of any available transportation asset.

- *Missions* - Mission objects provide a specification of a plan template (or basic plan class) for instantiating the transport plans that must be scheduled. In the strategic deployment domain, for example, the basic plan class corresponding to an individual TPFDD record is specified as an aggregate transport operation (at some level of precision with respect to required asset capacity), which decomposes into a load, travel, unload operation sequence.

Through definition of more specialized object classes, the constraints specified by any of these modeling primitives can be straightforwardly customized. For example, in modeling the IFD2 MEDCOM scenario in terms consistent with PFE (see discussion of experiments below), a specialization of transport operation was defined to incorporate the PFE definition of required capacity as a function of both commodity and asset type. The current DITOPS library of modeling primitives consists of 80 core (i.e., domain independent) classes and 40 additional specializations defined for specific application to military transportation planning domains. Full details of these primitives and their protocols may be found in [LSS+93].

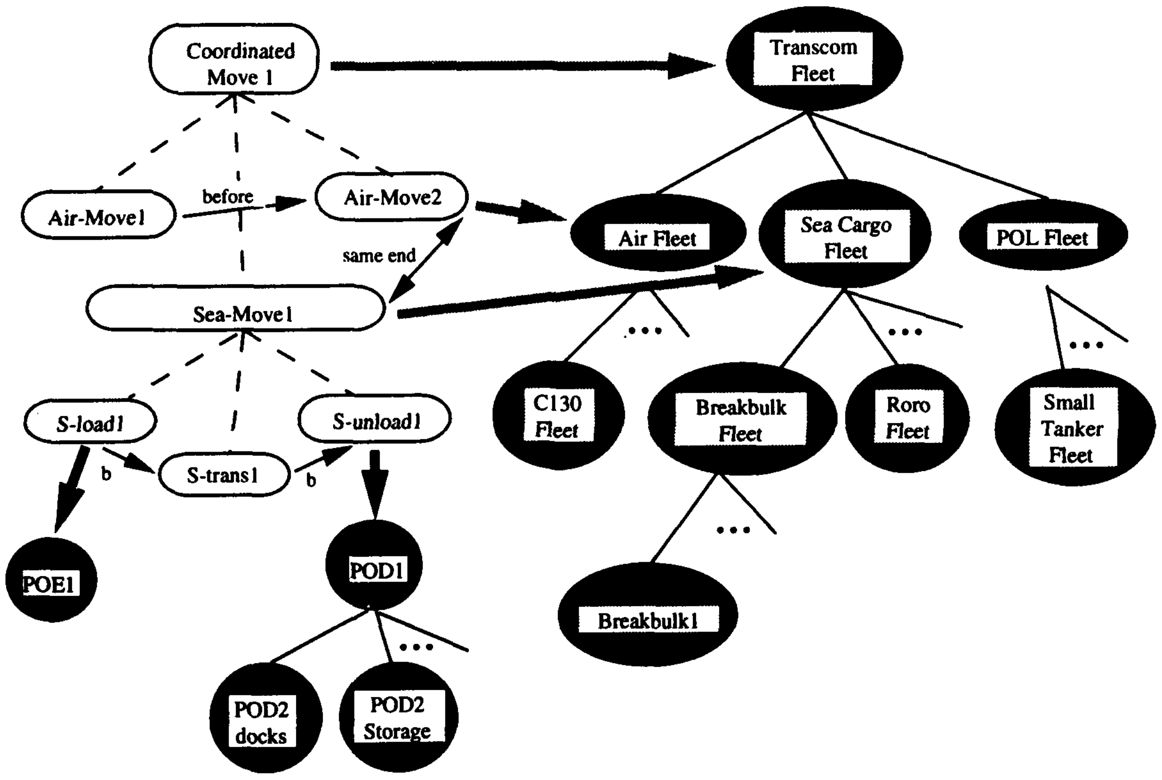


Figure 1: Hierarchical domain models

To give a flavor of the modeling capabilities provided by the DITOPS class library, we consider aspects of the domain model that was constructed for MEDCOM strategic deployment scenario just mentioned. Figure 1 graphically illustrates the defined hierarchical models of required resources and transportation processes.

In the resource model depicted, individual transportation assets are first composed into disjunctive aggregate resources (or resource pools) representing fleets of specific craft types. These descriptions, in turn, are aggregated into larger disjunctive aggregates representing higher-level pools of allocation alternatives (e.g., cargo/pol sea lift capacity, air/sea lift capacity). In this case, capacity constraints are straightforward mapped by summing the unit capacities of individual resources[1]. Representations of port resources are specified similarly.

[1]Mapping of other constraints (e.g., operating speeds, allowable cargo types) is accomplished by various approximation methods (e.g., weighted averaging, set union).

However, in this case the individual resources associated with a given port (e.g., loading unloading equipment, cargo storage space, etc.) are composed into a conjunctive aggregate resource, which provides a single, higher-level estimate of overall port capacity[2] On one hand, these levels of resource description define multiple levels of possible scheduling precision. For example, in computing transport mode assignments relative to an apportioned set of resources, there is likely little leverage to be gained by computing schedules at the level of individual resources. Alternatively, a level of scheduling precision appropriate for transportation feasibility analysis at the level of USTRANSCOM would include individual craft assignments but only aggregate accounting of port capacity constraints. Detailed models of atomic port resources would, however, become necessary at the stage of detailed execution planning. Having fixed a given level of scheduling precision (say individual craft assignments and aggregate accounting of port capacity), the hierarchical model additionally provides a structure for elaborating the search for a schedule. Summarized allocation constraints and preferences associated with aggregate resources (e.g., current available capacity, usage restrictions, preferred sub-resources) provide a basis for restricting and biasing consideration of resource alternatives.

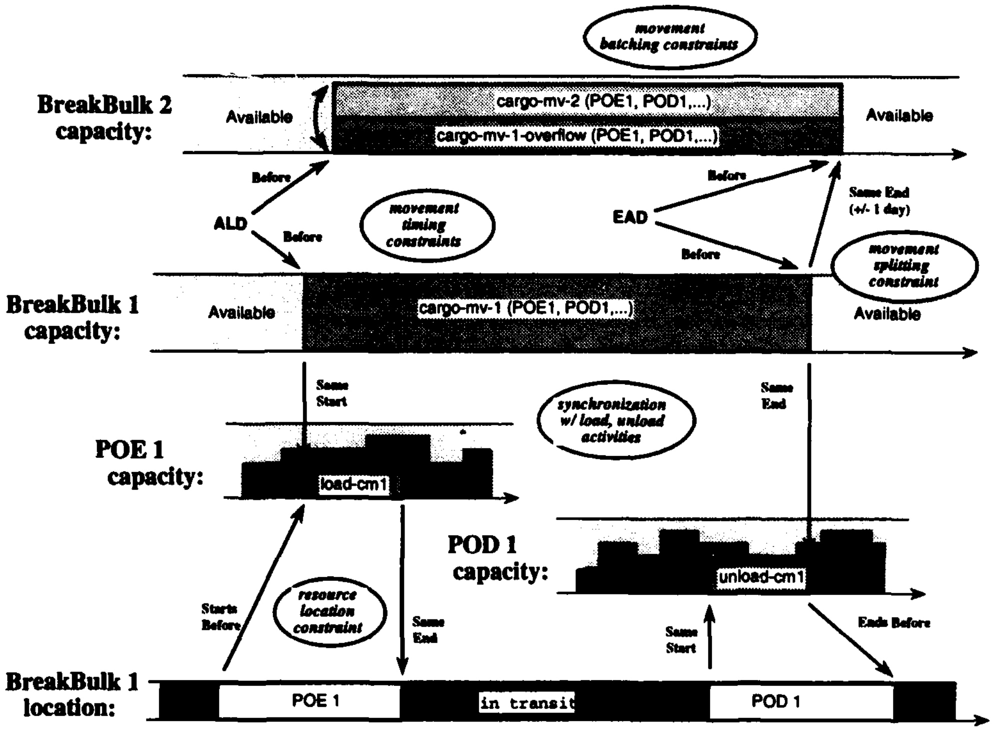


Figure 2: Allocation constraints in "MEDCOM" model

Figure 2 graphically illustrates the range of allocation constraints incorporated within the

[2]Several alternative mappings of capacity are possible here. The mapping might omit all but the most constraining atomic capacity; alternatively, the mapping might compute an overall throughput summary estimate. In some problem contexts, it may not be necessary, desirable or feasible to account for and model individual port capacity constraints, in which cases, aggregate port descriptions simply constitute the most detailed level of the model.

MEDCOM domain model. The figure centers around a particular scheduled transport activity, *cargo-mv-1*. It depicts the available capacity over time of two ships, ***breakbulk-1*** and ***breakbulk-2***, and two seaports, *POE-1* and *POD-1*, as well as the current location of ***breakbulk-1*** over time. The "box" labeled with *cargo-mv-1* within the available capacity profile of ***breakbulk-1*** represents the interval over which *cargo-mv-1* is scheduled to occur; since ***breakbulk-1*** is required and the quantity of *cargo-mv-1* fully consumes the capacity of the ship, it is unavailable for other use over this period (according to the depicted profile, it is currently available both before and after this scheduled trip). The activity *cargo-mv-1* abstracts a more detailed sequence of ***load-cm1***, ***transport-cm1*** and ***unload-cm1*** operations. Both the ***load-cm1*** and ***unload-cm1*** operations additionally require port capacity at *POE1* and *POD1* respectively. According to the defined hierarchical model, ***load-cm1*** is constrained to commence at the beginning of the overall *cargo-mv-1* operation (i.e. at the same point that ***breakbulk-1*** is first allocated to *cargo-mv-1*), and, conversely ***unload-cm1*** is constrained to end coincident with the "release" of ***breakbulk-1*** by *cargo-mv-1*. During the scheduled interval of both ***load-cm1*** and ***unload-cm1***, the required amount of port capacity (in this case a function of the cargo quantity) is designated as allocated to these operations and otherwise unavailable. Execution of either ***load-cm1*** or ***unload-cm1*** also requires the transport resource to be physically present at the port. These constraints are specified in the model as operation "setup" constraints that must be satisfied, and are enforced by ensuring that the resource is at the designated POE at the scheduled start of any load operation. If , during scheduling of a load operation the assigned transport resource is not at the load site, checks are made to ensure that the resource is available sufficiently earlier than the scheduled start to enable it to travel to the load site.

The example depicted in Fig' ^ 2 also illustrates three other types of allocation constraints that are specified in the MEDCOM model and taken into account when scheduling transportation activities. In this example, *cargo-mv-1* actually moves only a portion of the cargo designated in its associated move requirement (i.e., the TPFDD record that led to creation of *cargo-mv-1* in the first place). Since the maximum carrying capacity of ***breakbulk-1*** is not sufficient to accommodate the entire input requirement, the load has been dynamically "split" into two loads. A second *cargo-mv-1-overflow* activity has been created and scheduled on a different resource ***breakbulk-2***. In this case, both *cargo-mv-1* and *cargo-mv-1-overflow* share the common ALD (available to load) and EAD (earliest arrival date) constraints on scheduled start and end times that are specified in the associated move requirement. Whenever the cargo associated with a given move requirement must be split across multiple trips, a default constraint on their relative timing is also imposed. In this example, the two transport activities are constrained to finish within one day of each other. Finally, it is often the case that the capacity of a given transportation resource is sufficient to simultaneously support multiple transport activities, transporting their respective cargos as a "batch" on the same trip. This is the case for the ***breakbulk-2*** trip that is depicted in Figure 2 where *cargo-mv-1-overflow* has been batched with second transport activity *cargo-mv-2*. For transport activities to be batched, several constraints may have to be satisfied. Minimally, the activities must have the same designated POEs and PODs. Although not specified in the MEDCOM model, additional constraints relating to the compatibility of different cargo types might also be defined and enforced.

3.2 Building and Managing Transportation Schedules

Scheduling in DITOPS is formulated as a reactive process, reflecting the fact that a schedule at any level or stage of the deployment planning process is a dynamic evolving entity, and is continuously influenced by changing mission requirements, conflicting decision-making perspectives/goals and changing executional circumstances. This problem solving perspective in large part motivates the above illustrated representations of changing resource state over time (i.e., available capacity, location). These representations are pre-requisite to the specification of procedures for reflecting the consequences of changed constraints and for incrementally managing schedules in response to such changes. These representations also enable use of schedule building and revision procedures other than time-forward simulation, which is inherently myopic and susceptible to sub-optimal decision-making.

Most generally, the DITOPS scheduling model can be seen as a constraint-based scheduling model; instantiated movement plans define sets of start/end time and transport resource decision variables, and decision-making is concerned with establishing (or restoring) an assignment of times and resources to all variables that is consistent with specified temporal synchronization and resource utilization constraints. A constraint-based scheduling model is broadly characterized as an iterative procedure that combines three basic elements:

1. deductive constraint propagation techniques, which are applied to incrementally update the domains of decision variables in an underlying solution constraint graph as changes (or extensions) are made to the schedule and recognize inconsistencies,

2. look-ahead analysis techniques, which estimate the critical tradeoffs (decisions) and opportunities (flexibilities) implied by current solution constraints for purposes of determining which decision (or set of decisions) should be considered next, and

3. a decision procedure, or set of procedures, for carrying out specific solution changes or extensions.

In fine granularity scheduling models (e.g., [Sad94]), the look-ahead analysis and decision procedures map directly to the variable and value ordering heuristics of traditional constraint satisfaction problem solving procedures. DITOPS, alternatively, implements a "coarser granularity" model [Smi94]. Look-ahead analysis is instead used as a basis for heuristic problem structuring and subproblem formulation, which involves selection of a particular set of decision variables to focus on (i.e., assign or revise) and selection of a particular decision (or local search) procedure to apply to this set of decisions. In either type of model, preference or utility structures (e.g., reflecting objective criteria and preferences) can be associated with decision variable values to bias the overall search process. In the case of DITOPS, alternative decision-making procedures are specifically designed to provide differential optimization and conflict resolution capabilities. In the absence of explicit user guidance, control heuristics which map analyses of the current solution state to important optimization (or reoptimization) needs and opportunities are used to opportunistically select the most appropriate decision procedure on each control cycle. For example, suppose a capacity conflict has been introduced into the schedule of a particular cargo ship due to it having been temporarily disabled. Activities scheduled over the expected period of unavailability must now be reassigned to other ships and loss of transport capacity implies the need to (re)optimize existing ship capacity to maximize utilization; a decision procedure with this optimizing property is the preferred procedure to apply. At the same time, reconsideration of the schedules of other ships capable of carrying the now stranded cargo should take into account current flexibilities in the solution. If in examining the available capacity of the fleet to which the failed ship belongs it is estimated that sufficient extra capacity to resolve the problem, there is no reason to consider any other viable resource alternatives; the scope of the change is restricted to this smaller set of resources. Within DITOPS, this subproblem formulation activity is carried out by a designated procedure referred to as the top-level manager. The underlying system control architecture is graphically shown in Figure 3.

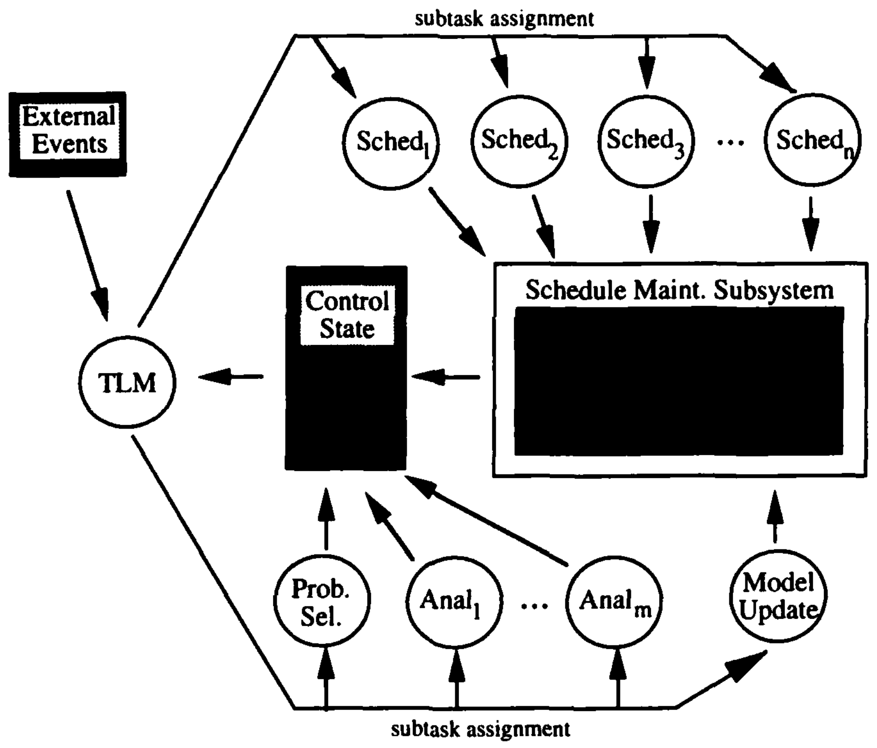


Figure 3: The DITOPS Scheduling Architecture

Two constraint analysis procedures are available within the DITOPS scheduler to support the control decisions of the top-level manager. In situations where scheduling decisions remain to be made, a capacity analysis procedure provides estimations of likely resource bottlenecks. In situations of detected constraint conflicts, a conflict analysis procedure computes a set of metrics, some of which estimate the severity of the problem and some of which characterize the looseness or tightness of time and capacity constraints in the local "neighborhood" of the schedule that contains the conflict.

A number of decision-making procedures are available for application in different scheduling or rescheduling contexts. Local search methods are defined for both "resource" and "movement" centered scheduling, providing capabilities, respectively, for manipulating (i.e., revising or extending) the schedules associated with particular sets of resources (e.g., the cargo ship fleet) or particular sets of temporally related movements (e.g., the movements associated with a particular force module). By virtue of search orientation, each of these methods emphasizes specific optimization biases; resource scheduling promotes efficient use of available transport capacity while attempting to minimize the tardiness of scheduled movements. Movement scheduling, alternatively, promotes enforcement of arrival constraints and efficient synchronization of dependent movements, while attempting to minimize asset capacity requirements. Both of these methods share a common search infra-structure that

- incorporates machinery foı incrementally propagating consequences of scheduling decisions and detecting constraint conflicts (referred to as the "schedule maintenance subsystem" in Figure 3),
- provides primitives for generating feasible decision alternatives (based on the use of aggregate resource and activity descriptions defined in the underlying domain model), and
- allows incorporation of additional allocation preferences, which are expressed in the domain model as utility functions over the possible values of decision variables (e.g., possible resource assignments, possible activity start times) and integrated as terms of the search procedures' evaluation function.

A number of more specialized revision procedures have also been defined, providing additional capabilities to shift the scheduled interval of scheduled "trips", to swap scheduled batches of particular transportation assets, and to balance cargo load to exploit increases in port capacity. The search infra-structure and decision procedures are defined and implemented compositionally using object-oriented techniques, providing a functional "tool box" for constructing additional decision-making procedures.[SL93]

3.3 Experimentation and Performance Analysis

In demonstrating and evaluating the capabilities of the DITOPS transportation scheduler, we have focused principally on the strategic deployment planning task addressed by the US Joint Transportation Command (USTRANSCOM). At this level of the logistics planning process, planning is concerned with the development, analysis and management of a Time-Phased Force Deployment Database (or TPFDD), which specifies the complete set of personnel and cargo movements required to support a given employment plan and all associated deployment constraints (e.g., earliest/latest departure and arrival dates, transport modes, origins and destinations, etc.). We have utilized a representative TPFDD provided within the ARPA PI CPE (referred to as the MEDCOM scenario) to demonstrate a range of decision support and decision making capabilities provided by the DITOPS transportation scheduler. We summarize this work in the subsections below.

Item	Description
POE	Port of embarkation (origin)
POD	Port of debarkation (destination)
ALD	Available to load date
EAD	Earliest arrival date
LAD	Latest arrival date
cargo-type	categorization of type of cargo used to formulate usage (or carrying) constraints for different resource types. One of: 'bulk outsize, oversize, pax, nat, pol, container, roro, breakbulk'
commodity-type	finer categorization of cargo type used to determine capacity requirements on feasible asset types. One of: 'airborne, air-cavalry, air-mobile, armored, infantry, mechanized, combat-support, combat-service-support, navy, air-force, marines, resupply, ammunition, pol, pax'
MTONS	sea cargo quantity in metric tons (weight)
STONS	air cargo quantity in short tons (volume)
PAX	quantity for PAX cargo - number of passengers
CBARRELS	quantity of POL cargo - number of 100 barrels

Figure 4: TPFDD Record Information

3.3.1 Generating TPFDD level schedules

The principal task supported by current scheduling tools at USTRANSCOM is *transportation feasibility analysis*: given a fully specified TPFDD and a profile of apportioned sea and air lift assets, generate a deployment schedule that assigns personnel and cargo to be moved to specific lift assets over time in accordance with specified constraints. To assess the capabilities of DITOPS in this capacity, we conducted a comparative experiment with a BBN developed feasibility estimator called PFE. PFE is a simulation-based technology based directly on the now operational DART simulation tool, and is quite representative of the tools currently in use at USTRANSCOM.[SMS+91]

The experimental comparison was carried out using the MEDCOM TPFDD that was generated during the course of the 2nd PI Integrated Feasibility Demonstration (IFD2). This TPFDD contains a total of 3001 movement requirements, of which 1187 are pre-designated as air movements and 1814 are pre-designated as sea movements. The information provided with each movement requirement is listed in Figure 4. Sea movements can be further decomposed into 1323 sea cargo movements (requiring capacity on some subset of five different types of cargo carrying vessels) and POL movements (requiring capacity on oil tankers). Given the pre-assignment of transport mode and the absence of temporal constraints on the relative timings of various sea cargo, air, and pol movements in this scenario, the problem is decomposable into 3 mutually exclusive subproblems. Air and sea assets apportioned to support the deployment consisted of 369 aircraft, 36 cargo ships, and 4 tankers, with initial locations and staged availability as indicated in Figure 5.

Craft Type		Total Number	Initial Location	Availability							
				C0	C1	C2	C3	C4	C5	C7	C10
Air											
	C5	56	TMKH	56							
	C130	128	TMKH	128							
	C141B	185	TMKH	185							
Sea Cargo											
	FBB	20	BBNV					6			
			DKSD	8	3			1			2
	FLASH	6	BBNV				2				
			DKSD		3		1				
	FRORO	6	BBNV		1		1				
			DKSD	1	1		2				
	FSSC	3	DKSD		2			1			
	FSEAB	2	DKSD		1		1				
Sea POL											
	SMTNK	2	DKSD		1		1				
	MDTNK	1	DKSD				1				
	LGTNK	1	DKSD							1	

Figure 5: MEDCOM scenario lift assets and availability

In collaboration with BBN, a model of scenario resources and resource utilization/allocation constraints equivalent to that employed in PFE was configured and instantiated. In particular, asset usage constraints were defined by associating a specific subset of allowable cargo types with each type of craft (e.g., C141Bs can only carry 'bulk' cargo). Asset capacity constraints were specified for each asset type as a vector of (commodity type quantity) pairs, over which capacity requirements for a movement of a specific commodity type on a specific type of resource were formulated as a function of the percentage of the resource required. Availability and locations of specific transportation assets were initialized according to the constraints in Figure 5, and travel times were based on identical models of resource operating speeds and inter-port distances. Equivalent port throughput capacity constraints were specified, including a reduced throughput capacity of 50,000 Mtons/day at one POD (Tunis) which was called for in the scenario to introduce greater congestion. There was one point of difference between the PFE and DITOPS models with respect to modeling both port and air-lift capacity. In DITOPS, capacity constraints were defined with respect to continuous time (i.e., how much capacity is available at any point in time), while PFE relies on less precise "capacity per day" models. In this regard, load and unload durations were assumed to be one day each for sea movements (consistent with PFE) and one hour each for air movements (below the granularity of the PFE simulation). Complete details of all port and asset capacity constraints can be found in the CPE description of the MEDCOM scenario.

In conducting the experiment, we focused on three dimensions of system performance:

1. the ability to enforce important deployment constraints - this dimension concerns the

System/Config.	Air Movements (1187 total)	Sea Cargo Mvmts. (1323 total)	POL Movements (491 total)
PFE			
w/o EAD enforc:	% tardy: 0	% tardy: 84	% tardy: 59
DITOPS			
w/o EAD enforc:	% tardy: 0	% tardy: 58	% tardy: 1
w/ EAD enforc:	% tardy: 0	% tardy: 78	% tardy: 90

Figure 6: Comparative performance of DITOPS and PFE

reliability of the schedule as an indicator of deployment feasibility. In the case of the MEDCOM scenario, earliest arrival date constraints (EADs) were intended to be enforced as hard constraints (to preserve the element of surprise), but this was not possible withing PFE. We conducted runs with DITOPS with and without the assumption that EADs should be enforced as hard constraints, to demonstrate the potential variance in results.[3]

2. ability to optimize with respect to important deployment objectives - this dimension measures the system ability to produce better quality schedules, and hence better guidance to more detailed (e.g., component command) planning processes. Here our principal measure of performance was level of tardiness observed in the 'closure' of various movements under both generated schedules. We also tracked resource utilization over time, but due to the nature of the PFE simulation (as run by BBN), comparison was not possible along this dimension.

3. the computational cost of the scheduling process - the issue here is system efficiency and scalability.

Comparative results with respect to tardiness on the IFD2 problem are given in Table 6. As can be seen, there is sufficient air lift capacity to meet movement delivery dates (LADs) and the deployment schedules of both PFE and DITOPS show no tardiness. The situation is different with respect to the sea transport portion of the problem. With respect to sea cargo, for example, DITOPS produced a deployment schedule with a 6% reduction in late closures over the PFE schedule. Average resource utilization by resource type ranged from 51% to 100%, with an overall average utilization of 85%. It was not possible to compute and compare average tardiness figures, since the PFE simulation apparently terminates after 70 days (in this case, failing to schedule 32% of the movements).

This reduction in tardiness is significant for a couple of reasons. First, it was achieved while enforcing EAD constraints that were ignored by PFE and adversely affect the scheduler's ability to minimize late closures. A run of DITOPS where these constraints were also ignored (also included in Table 1) yielded a 26% reduction in tardiness. The results are even more dramatic in the case of POL movements, where, without EAD constraint enforcement,

[3]In actuality, it is not the case in this scenario that all EADs are in fact hard constraints; however, since information needed to differentiate was not available we assumed the worst case.

DITOPS produces a schedule with only 1% of the movements tardy as compared to PFE's schedule with 59% of the movements tardy. Second, these initial results were obtained with fairly generic scheduling methods. We expect even better results as heuristics that further exploit the structure of the problem are incorporated.

Using the ported CommonLisp/CLOS system, the schedules reported above are generated in just over 10 minutes on a SUN Sparcstation 10, indicating the ability of the DITOPS scheduler to scale to realistically sized problems. Experiments have also been performed under assumptions that port capacity constraints are not limiting and can therefore be ignored (which, for example, matches the modeling assumptions of the Kestrel scheduler[Smi92]). If the port capacity constraints specified for the MEDCOM problem are ignored, schedule generation time is reduced to about 7 minutes.

3.3.2 TPFDD Mode Assignment

We have also conducted experiments to demonstrate capabilities in support of decisions made earlier in the deployment planning process that in current practice are made without consideration of resource capacity constraints. We have performed preliminary experiments using (a variant of the IFD2 scenario) that demonstrate the potential impact of basing "transport mode" decisions on resource capacity information. Specifically, the mode decisions designated in the input IFD2 TPFDD were stripped off, and, using the constraints relating to various asset capabilities and cargo commodity types, aggregate level schedules were generated which assigned either air or sea lift capacity to specific move requirements. The results obtained varied considerably from the original mode assignments, exploiting the excess air lift capacity implied by the detailed TPFDD scheduling experiments described above. Although it is not clear whether represented resource usage and cargo commodity constraints are sufficient alone to determine feasible air and sea assignments, in this case, the redistribution of mode assignments resulted in better closure profiles.

3.4 Interactive/Reactive Schedule Revision

The reactive scheduling framework of DITOPS provides equally important capabilities for incrementally revising schedules, either in response to changes in external circumstances (e.g., the unexpected fog-in of a port or the receipt of additional deployment requirements) or for purposes of improving a schedule with observed deficiencies (e.g., by apportioning additional transport resources). From a mixed-initiative scheduling perspective, this reactive framework promotes a default style of interaction grounded in user manipulation of problem constraints (e.g., resource capacity and availability, activity deadlines, etc.) and system determination of consequences (using internal strategies for reconciling conflict resolution and solution improvement possibilities with the desire to minimize schedule disruption). Though this division of responsibility may match user decision-making goals in some cases, it will more frequently be the case that realization of system activity consistent with user expectations will necessitate greater user involvement in the system's subproblem formulation

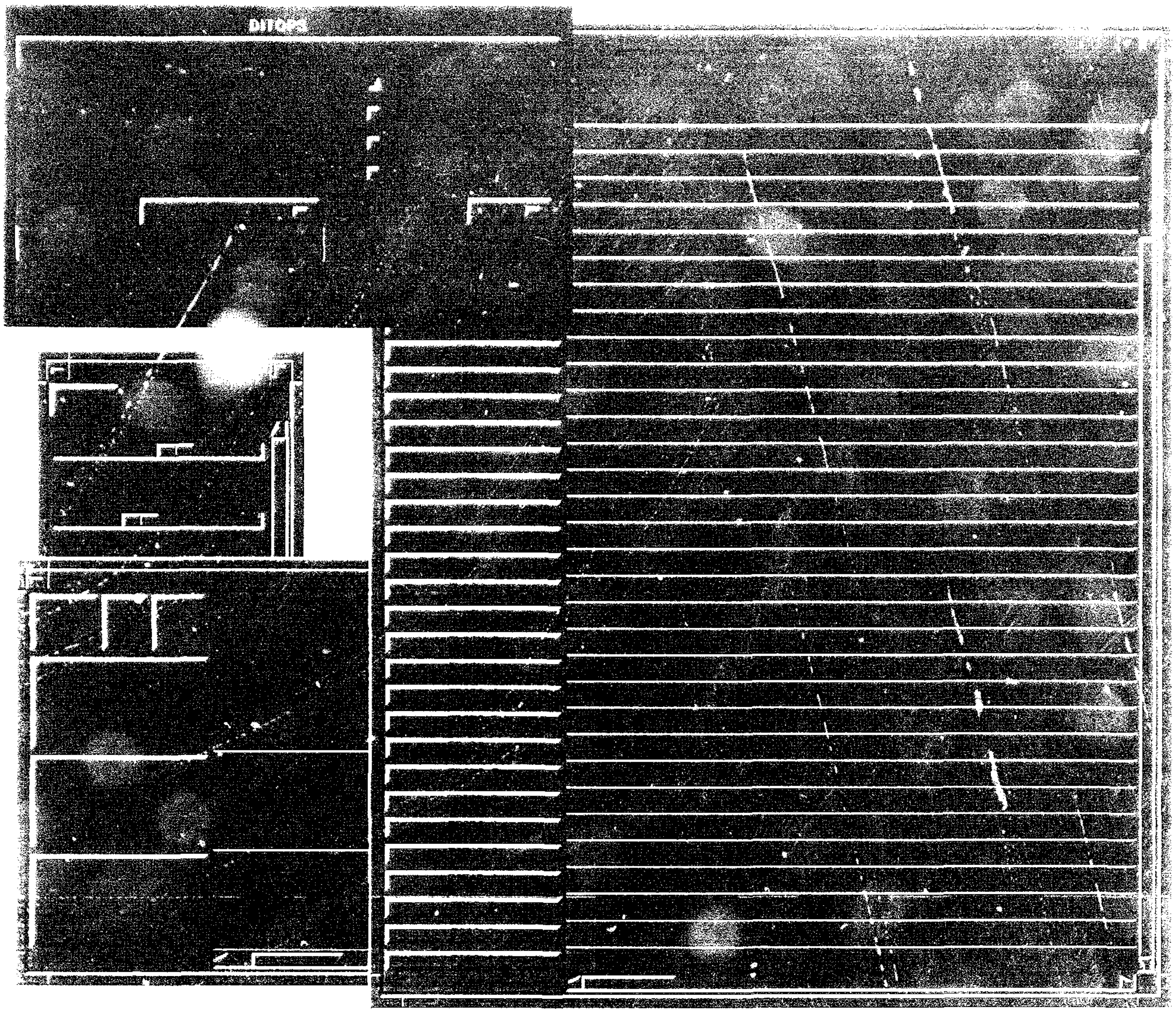


Figure 7: The DITOPS User Interface

Interaction between the user and the current DITOPS scheduler occurs through a graphical direct manipulation interface which emphasizes visualization and manipulation of schedules in terms of resource capacity utilization over time (see Figure 7). Based on the underlying hierarchical resource model, the user can create resource capacity views at various levels of aggregation. This allows the user to examine either individual craft assets, fleets or ports. The resource capacity views support zooming and scrolling for localizing attention on particular resources and/or regions of the overall schedule horizon. The user can select temporal intervals by "boxing" the area of interest with the mouse. Any querying and manipulation of schedules and solution constraints is based on these uniform time selections; once a selection has been made a variety of actions is possible through a menu associated with the resource in question.

Given a selected interval of time, the user may choose to examine properties of the delineated portion of the resource schedule. If the resource is an individual craft asset, for example, the transport activities supported by scheduled trips are accessible. At aggregate resource levels, graphical displays of various properties of the solution can be retrieved (e.g., movement closure profiles, accumulated cargo tonnage over time[4]). This provides a basis for identification of solution deficiencies.

User manipulation of problem constraints and schedules also centers around a selected resource profile interval. A transport or port resource can be made unavailable over a selected interval. In this case, any inconsistencies in the schedule that result are highlighted. Conversely, resource capacity of a given fleet can be increased for a specified interval by moving to the appropriate aggregate resource display (this translates to adding craft to the fleet). As indicated earlier, such a "relaxation" of capacity constraints should generally be accompanied by an indication of the action focus and scope (reflecting the specific rescheduling goal that motivates the change). Within the current implementation, only fixed choices relating to activities that are currently late and resource usage restrictions are available for narrowing system focus and scope. The "current time" indicator at the top of the resource displays can be moved along the schedule horizon to simulate states during the execution of the schedule. Default rescheduling biases are adjustable through a "slider" display which represents the relative importance to be attributed to each system known preference. In imposing any given change to the current schedule, there is no obligation to the user to provide additional revision constraints and guidance; generally speaking, user decisions along these lines are considered to be defaults until they are changed.

Overall system activity is managed through a "control panel" (upper left corner of Figure 7), which provides capabilities for creating various displays, loading scenario descriptions and deployment problems (sets of move requirements), saving and reloading generated schedules, and adjusting global system parameters and preferences (e.g., level of scheduling precision, automatic or selectable system response to changes, etc.).

3.5 Multi-Level, Distributed Scheduling

Transportation scheduling is inherently a distributed problem. Given its overall size and complexity, as well as the component structure of the military command, responsibility for different parts of the problem at different stages of the process are distributed among many

[4]In part, these capabilities draw on the SciGraph package.

planning agents. In current military transportation practice, schedules are produced by different agents along different lines of decomposition. For example, (a) different agents (e.g., CENTCOM, PACOM) produce schedules at the same level of aggregation for different military operations (e.g., multiple simultaneous crises), (b) different agents (e.g., USTRANSCOM, MAC, MITMIC) produce schedules for the same operational scenario at different levels of aggregation, or (c) different agents produce schedules for different resources (e.g., tankers, crews, cargo-handling equipment at a port). In all cases, resolution of conflicts is an integral issue. Although decomposition is an effective means of reducing problem complexity, effective and efficient decision-making requires mechanisms for coordinated interaction.

To support investigation into and experimentation with protocols and strategies for coordinating multiple scheduling agents, the DITOPS infra-structure also incorporates primitives for asynchronous communication. These primitives allow easy implementation of agents, their control architectures and inter-agent messages. In addition, some of the basic services utilized within the DITOPS scheduler (e.g., time services) are designed to allow experimentation in a distributed, asynchronous environment. The design of the DITOPS communication substrate, like the rest of the DITOPS system, relies heavily on object-oriented programming concepts, and is influenced by earlier work reported in [LT91].

This system base has been used to define and implement an initial prototype system for distributed, multi-agent scheduling. We summarize the basic properties of the model underlying the prototype and the demonstration experiment that was performed below.

3.5.1 Decomposition and Interaction Assumptions

The hierarchical descriptions of resources and resource constraints advocated by the DITOPS modeling framework provide a natural basis for decomposing and structuring solutions to the overall transportation planning/scheduling problem. As previously observed, they provide a basis for specifying schedules at different levels to support decision-making at different stages of the planning/scheduling process. They likewise provide a structure for decomposing and distributing problem solving responsibility, where different agents are responsible for allocation/apportionment of specific sets of resources at a given level of detail (e.g. overall transport capacity, sea/air transport assets, port resources). Building from this basic problem decomposition perspective, we have developed a specific model for distributed, multi-level generation and management of transportation schedules. The model assumes a hierarchical organization of scheduling sgents, with each agent having access to specific levels of underlying hierarchical domain model (in effect, the "full" hierarchical model is distributed among scheduling agents). Thus, there is heterogeneity in the portion and level of description of the overall problem accessible to each agent.

Given the scale of the overall problem and the use of abstractions of resource allocation constraints as a basis for specifying problems and solutions at different levels, two further decomposition assumptions follow directly:

- **Decision-making scope and granularity** - The portion of overall problem that is visible and of concern to the decision-maker and correspondingly the level of detail of supporting models can be seen in relation to particular stages of the overall process. For example, transportation feasibility analysis during course of action development requires a global (and necessarily coarse) view of the whole problem. Management of day to day activities at a port, alternatively, requires much more detailed models of temporal process constraints and resource constraints, but only with respect to activities surrounding the use of the port.
- **Horizon of decision-making** - Corresponding to decreasing scope and increasing model detail is a decrease in the temporal horizon of decision-making. This assumption is supported by two considerations: problem scale and presence of environmental uncertainty. The problem solver's computational burden can remain almost invariant at each level by balancing decreasing scope and increasing model detail. The extent of uncertainty in the operating environment makes the executability of more detailed models more suspect further into the future. Thus a given decision-maker's horizon must balance the computational burden of maintaining the solution over time (or equivalently the extent to which it really provides a useful projection of future events)

These collective assumptions lead to a distributed model that resembles the organization and roles of current transportation planning command and control structures. This is illustrated in Figure 8.

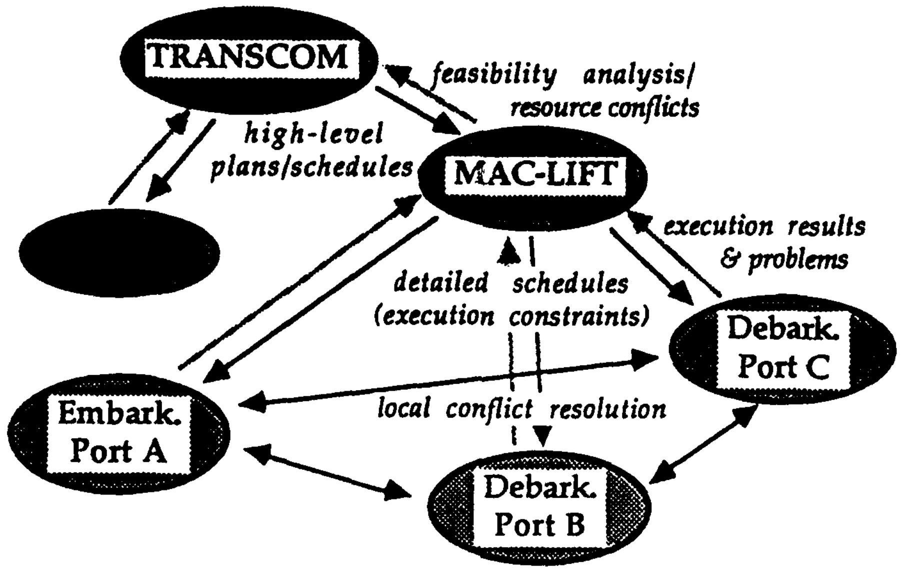


Figure 8: Distributed, multi-level transportation scheduling

Within this model there are two basic types of agent interactions:

- **Vertical**: The results of a given agent's scheduling (or rescheduling) actions are communicated downward as scheduling constraints/objectives; an agent's ability to satisfy imposed constraints/objectives, or responses to lower-level results are communicated upward. At each level of abstraction, an agent produces the best solution it can, given currently imposed global/constraints and objectives and the currently known results communicated from lower level agent results (or the execution environment)
- **Lateral**: Agents at the same level communicate to resolve local conflicts and produce solutions within bounds of constraints that have been imposed through downward constraint communication.

Coordination of the overall organization of agents is achieved by the following "interaction policies":

- Each agent is responsible for generating scheduling constraints for the agents directly under it (in the subtree of which it is the root). At the same time, since a lower level agent has a more detailed model of its own activities than a higher level agent has of it, the lower level agent can react more effectively to schedule deviations that are encountered at its level. Hence reaction starts at the level where the schedule deviation occurs and its effects are propagated both downwards (in terms of new constraints) and upwards (in terms of violations of imposed constraints that may result in potential rescheduling decisions at a higher level).
- Deviations will always be responded to locally to the extent possible (engaging other local agents as necessary) - i.e. there must always be the ability to drive execution without communication to the "superior" agent,
- (re)scheduling results produced in response to deviations are always communicated upward if previously imposed guidelines (constraints) have been broken (and of course always propagated downward). If it can be recognized without local schedule revision that the deviation will break imposed constraints (e.g. a port fogin), then the deviation can be communicated upward immediately. In this case, a local agent must still try to make do (resolve problems) until its superior agent responds,
- If it becomes time to act (either in response to execution demands or to respond to lower level agents), then whatever current local solution exists is followed.
- If a superior agent, in response to either deviation or revised solution received from below, revises its more global solution, then revised constraints/guidelines are communicated downward to its inferior agents. Assuming a cooperative framework, these new constraints are given priority in resolving conflicts (i.e. if the revised constraints are inconsistent with current inferior agent schedules, then the inferior agent is obliged to revise). One issue that arises in this protocol is the detail/granularity of the superior agent's model (and the possible mismatch with the inferior agent's more detailed model). However, if this is ultimately the reason to prefer an inferior agent's solution,

then that will subsequently be discovered by its inability to meet the newly imposed constraints - in which case the best solution that the agent can produce is communicated upward.

3.5.2 Distributed Experiments

To demonstrate the above framework for multi-level transportation scheduling, a simple system configuration consisting of a single high-level (i.e., TRANSCOM-level) agent and two more detailed port scheduling agents was implemented and tested. The high-level agent was responsible of generating a deployment schedule for movements from port A to port B (under the aggregate port capacity models utilized in the DITOPS TPFDD scheduling experiments). Both port A and port B were assumed to have their own scheduling agents, whose responsibility was to develop more detailed schedules, involving allocation of constituent port resources (docking berths, loading equipment, etc). The high level agent operated with an overall horizon matching the total duration of the deployment at a temporal granularity of days, while the low-level port agents scheduled over a shorter horizon, defined relative to the travel time required for transport assets to move from port A to port B, and and produced hourly schedules.

In the scenario demonstrated, the high-level agent would generate an initial deployment schedule and communicate these results "downward" to port agents, requesting them each to generate a port schedule for the designated movements, given arrival and departure dates based on the high-level agent's schedule. The port agents would then generate a schedule for their own resources. Finding this impossible, the port agents would communicate with each other, possibly requesting arrival and departure dates to be shifted to arrive in a feasible solution. Once feasible detailed schedules were obtained, the port agents would then communicate their results "upwards" to the high-level agent.

Initially this scenario was simulated in a simple single-process environment. After developing an understanding of the necessary message types required for this type of distributed scheduling, the scenario was converted to function in a multi-process, multi-agent environment. This work has led to the design and implementation of a class library for asynchronous agents, providing primitives for the construction of the internal structure of agents (messages, message queues), agent control architectures (event processing mechanisms, tasks) and low-level services (network communication services, time and synchronization services). Further details of these mechanisms can be found in [LSS+93].

References

[LSS+93] O. Lassila, S.F. Smith, K. Sycara, G. Amiri, M. Becker, and C. Young. The ditops design reference manual. Technical report, The Robotics Institute, Carnegie Mellon University, September 1993.

[LT91] O. Lassila and S. Torma. Using a distributed frame system to implement distributed problem solvers. Technical Report HTKK-TKO-B68, Department of Computer Science, Helsinki University of Technology, Finland, 1991.

[Sad94] N. Sadeh. The micro-boss factory scheduling system. In *Intelligent Scheduling*, chapter 4. Morgan Kaufmann Publishers, Palo Alto, CA, in press, 1994.

[SL93] S.F. Smith and O. Lassila. Configurable systems for reactive production management. In *Proceedings IFIP TC 5 / WG 5.7 International Workshop on Knowledge-Based Reactive Scheduling*, Athens, Greece, October 1993.

[Smi92] D.R. Smith. Transformational approach to scheduling. Technical Report KES.U.92.2, Kestrel Institute, December 1992.

[Smi94] S.F. Smith. Opis: A methodology and architecture for reactive scheduling. In *Intelligent Scheduling*, chapter 2. Morgan Kaufmann Publishers, Palo Alto, CA, in press, 1994.

[SMS+91] J. Schank, M. Mattock, G. Sumner, I. Greenberg, J. Rothenberg, and J.P Stucker. A review of strategic mobility models and analysis. Technical Report R-3926-JS, Rand Corporation, 1991.

4 Additional Technology Integration Experiments and Support Services

In addition to the tpfdd-level scheduling capabilities summarized above, component scheduling "services" were also configured to provide constraint analysis support for higher-level course of action (COA) development:

- In collaboration with SRI, the resource capacity analysis capability used within the DITOPS scheduler was adapted for integration with the SOCAP planning system to provide feedback on transportation feasibility during generation of the deployment actions required to support the COA.
- Adapting component constraint propagation and conflict analysis techniques utilized within the DITOPS scheduler, a COA feasibility checker was developed and exported for incorporation into the TARGET IFD3 planning system. designed to verify consistency of the temporal constraints and force assignments in a given employment plan, and identify the set of conflicting constraints in inconsistent situations.

These auxiliary subsystems are summarized in the following two subsections. Complete details may be found in the DITOPS Design Reference Manual [LSS$^+$93].

4.1 Integrating Capacity Analysis into SOCAP

The DITOPS capacity analysis service was developed to show (in collaboration with SRI) the utility of integrating resource contention analysis into higher level COA planning and was demonstrated within the SOCAP planner. It was constructed as a direct extension of the capacity analysis procedure utilized within the DITOPS scheduler itself. In brief, this base procedure operates by first computing an infinite capacity schedule (i.e., a schedule in which all temporal constraints are satisfied) at some specified level of time and resource granularity, and then relating the resource capacity required by the schedule to the amount of resource capacity that is actually available over time. Subintervals of the scheduling horizon in which the demand for capacity on some resource (or set of resources) exceeds the available supply are then identified and returned as likely scheduling bottlenecks. For use within SOCAP, a protocol for mapping COA plan nets into the DITOPS schedule representation (and likewise for communicating results back) was developed and integrated with the base capacity analysis procedure.

The capacity analyzer was incorporated into SOCAP as an additional "plan critic" for use after completion of the deployment planning phase of its overall COA generation process. Its use was demonstrated in the context of the original IFD2 (MEDCOM) problem scenario; upon generation of a deployment plan which assumed only a single in-theater POD, application of capacity analyzer was found to (correctly) identify the insufficiency of this one port assumption from the standpoint of required port throughput capacity. Information relating to this detected port capacity bottleneck subsequently resulted in the triggering of a plan revision process in SOCAP, wherein a second in-theater POD was added to the plan. The overall point illustrated in this technology integration experiment was that consideration of capacity constraints early on in the planning process can lead to early detection of problems that, in current planning practices, might only be discovered after the initial deployment plan had been "exploded" to the detailed TPFDD level.

The DITOPS capacity analysis service is currently installed as a CPE knowledge service. In brief, it accepts a (typically high level) deployment plan along with a specification of available resources and resource capacity (i.e., ship, plane, port) identified as required in the plan. It returns as output, a set of any "bottleneck" resource intervals, and a resource usage profile for each resource over the plan horizon. The plan is communicated as a list of activities and temporal constraints. Each input activity description (corresponding to a node in the SOCAP plan net) contains the transport resource or resource type it requires, its POE, its POD, its cargo quantities (in terms of stons, mtons and/or pax), and any current bounds on its start time, end time, and duration. Each temporal constraint description identifies a binary temporal relation (e.g., before, same-start), the two activities that are constrained by the relation, and any quantitative bounds on the relation. Available transport resources are also communicated, with each resource description designating both an asset type (e.g., tairlift) and a set of instances (e.g., (1st-C130 2nd-C130 ...)). An additional set of inputs corresponds to periods of reduced available capacity for a given resource (transport asset or port), with each description indicating a specific resource, the amount of capacity lost, and the start and end time of the reduced capacity interval. Finally, parameters which establish the desired temporal granularity of the analysis and the threshold to be used in detecting bottlenecks are passed. The usage (or capacity) profile that is computed for each resource is returned as an ordered list of capacity intervals of the specified granularity (e.g., 24 hours), each of which specifies a start time, an end time, the projected demand for capacity over the interval, the available supply of capacity over the interval, and the demand/supply ratio. Any subsequence of intervals with a demand/supply ratio greater than the specified threshold is returned as an identified bottleneck.

4.2 COA Feasibility Checker

The COA Feasibility Checker subsystem was developed for use within the IFD3 TARGET system, and is currently a functioning component of this system. Similar in spirit to the capacity analyzer integrated into the SOCAP system, it performs feasibility checks on COA plans that are developed interactively within TARGET. However, there are important functional differences. First, the problem context is employment planning as opposed to deployment planning. In employment planning, forces are interpreted as the resources required by plan activities and whose availability must checked. Second, the qualitative temporal constraints on plan activities provided to the feasibility checker were not assumed to be consistent (as was the case with communicated SOCAP constraints); one objective of the feasibility checker is to provide guidance to the human planner in generating these temporal constraints. Third, the objective is not to estimate resource contention per se, but to instead identify and isolate sets of conflicting constraints.

The COA Feasibility Checker integrates time bound propagation techniques defined within the kernel DITOPS infra-structure with newly developed extensions to recognize and diagnose specific types of constraint conflicts. When provided with an input plan from TARGET, a topological sorting procedure is used in conjunction with time bound propagation to first check the plan for the presence of *cycles*. Detection of a cycle in this case implies the existence of some set of inconsistent temporal relations (e.g., A before B, B before C, C before A). If detected, a characterization of this constraint conflict, including the set of temporal relations that are involved, are returned for use in highlighting to the user which constraints must be changed to achieve a feasible plan. In cases of a cycle-free plan, checks are also performed to detect *time bound violations*, which indicate that the metric constraints imposed on the plan (e.g., mission start and end dates, activity durations) are not feasible, and *resource availability violations*, which indicate situations where the resources required by an activity (in this case, forces) are not available during the activity's inferred time window. Upon detection of either type of conflict, a description identifying the activities and resources involved in the conflict is returned, again to provide guidance in directing the plan change process. If an input employment plan is found to be conflict free, then the inferred time bounds for each constituent activity are returned as output.

5 Scheduling by Precedence Constraint Posting

In this section we summarize our research into the development of a novel approach to scheduling - scheduling by precedence constraint posting, as Smith and Cheng (1993) have termed it - demonstrate the power of the approach through development and experimental analysis of several scheduling algorithms based on this constraint posting concept. The scheduling algorithms based on this concept construct schedules by establishing processing orderings between pairs of operations on each machine, as opposed to more traditional algorithms which construct schedules by *repeatedly searching for the best start time for each* operation. Earlier indications (Smith and Cheng 1993) in this work appeared to lend weight to our supposition that by approaching a scheduling problem as one to establish processing orderings, flexibility and deferred commitments with respect to resource allocation decisions can be maintained so that it is more likely that simple, local heuristics can yield scheduling performance which is more competitive than the current dominating techniques.

To demonstrate the power of the constraint posting scheduling concept, we'll address two scheduling problems in this section: (1) job-shop makespan scheduling; (2) job-shop weighted-tardiness scheduling. These problems are chosen mainly because they have been extensively studied in prior literature and there exist several very best procedures, which can be severed as evaluation benchmarks to our new developments.

In very general terms, the approximation algorithm based on this scheduling concept contains two major steps. In step I, we select one pair of operations which are to be processed on the same machine. Then, we determine their processing ordering or sequencing constraint in step II. We call this step "precedence constraint posting". After the posting of one precedence constraint, the current search space gets pruned and we continue to search for the best solution on the new space. Repeating these two steps alternately, accompanied with the cycle prevention mechanisms, finally, we can determine the processing orderings for all pairs of operations on every machine and, thereby generate a complete schedule for the problem. Given below is the algorithmic description of the concept:

Step 0. Initialization.

Step 1. If all pairs of operations are ordered, stop. Otherwise,

Step 2. Pick one unordered pair.

Step 3. Post the proper precedence constraint between this pair.

Step 4. Update the current search space.

Step 5. Go to Step 1.

The rest of the section is organized as follows. In Section 2, we formally define the notations and scheduling problems which are to be used and studied throughout the section. Then, we briefly review the PCP algorithm previously developed by Smith and Cheng (1993) in

Table 1: Notations used in the study

n	=	number of jobs
m	=	number of machines
r_i	=	ready time or release date for job i
d_i	=	due date or deadline for job i
D	=	a common deadline
w_i	=	weight for job i
n_i	=	total number of operations required for job i
o_{ij}	=	operation j for job i
p_{ij}	=	processing time of operation o_{ij}
m_{ij}	=	machine requested by operation o_{ij}
est_{ij}	=	earliest start time of operation o_{ij}
lft_{ij}	=	latest finish time of operation o_{ij}
x^+	=	$\max\{0, x\}$

Section 3. Based on the PCP algorithm, we present a new approximation algorithm, called MULTIPCP, for the makespan problem. Performance study by comparing MULTIPCP and Shifting Bottleneck on two sets of problems is also given in Section 4. Section 5 presents another new approximation algorithm, called WTPCP, for the weighted-tardiness problem. To evaluate the effectiveness of this new algorithm, we conduct computational experiments and compare it with several best-known dispatching rules on a set of problems randomly generated for a simulated shop with 10 machines and 200 jobs. Finally, conclusions are given in Section 6.

5.1 Notations and Problems

Table 1 lists the notations to be used in this section.

In a job-shop problem, n job are to be processed by m machines with the following assumptions: (1) the sequence of machines for each job is prescribed; (2) every job can be processed by only one machine and every machine can process at most one job at a time; (3) machine setup times are not considered; (4) once a machine starts to process a job, no interruption is allowed. Given these assumptions, the objective is to generate schedules which satisfy some constraints or which minimize some functions of the completion times of the jobs.

In the minimum makespan problem, the objective is to find a schedule which minimizes the makespan, the total elapsed time needed to process all jobs. The makespan, C_{max}, of a schedule can be defined as

$$C_{max} = \max_i\{C_i\},$$

where C_i is the completion time for job i.

In the minimum weighted-tardiness problem, the objective is to generate a schedule which minimizes the mean weighted-tardiness of the problem. Although there does not exist an universally accepted criterion for due-based performance in job-shop scheduling, mean weighted tardiness (MWT) has been one of the most often-used in literature. MWT of a schedule can be defined as

$$\frac{1}{n}\sum_{i=1}^{n} w_i max(0, C_i - d_i),$$

where n is the total number of jobs; w_i is the weight, d_i the due date, and C_i the completion time for job i, respectively.

5.2 A Review of The PCP Algorithm

Given the problem data of processing times, job routings, ready times, and deadlines, PCP first initializes the earliest start time est_{ij} and latest finish time lft_{ij} for any operation o_{ij} by the expressions:

$$est_{ij} = r_i + \sum_{k=1}^{j-1} p_{ik}; \tag{1}$$

$$lft_{ij} = d_i - \sum_{k=j+1}^{n_i} p_{ik}. \tag{2}$$

After the initialization of earliest start and latest finish times for all operations, PCP, then, applies a procedure previously developed by Erschler et al. (1976), referred to as *Constraint-Based Analysis (CBA)*, to exploit dominance conditions and prune the infeasible regions on the current search space. To summarize their basic idea, assume that est_{ij} and lft_{ij} designate the current earliest start and latest finish time respectively of a given operation o_{ij}. Then, for any unordered pair of operations, o_{ij} and o_{kl}, where $m_{ij} = m_{kl}$, we can distinguish four different cases:

1. If $lft_{ij} - est_{kl} < p_{ij} + p_{kl} \leq lft_{kl} - est_{ij}$ then o_{ij} must be scheduled before o_{kl} in any feasible extension of the current ordering decisions. (case 1)

2. If $lft_{kl} - est_{ij} < p_{ij} + p_{kl} \leq lft_{ij} - est_{kl}$ then o_{kl} must be scheduled before o_{ij} in any feasible extension of the current ordering decisions. (case 2)

3. If $p_{ij} + p_{kl} > lft_{kl} - est_{ij}$ and $p_{ij} + p_{kl} > lft_{ij} - est_{kl}$ then there is no feasible schedule. (case 3)

4. If $p_{ij} + p_{kl} \leq lft_{kl} - est_{ij}$ and $p_{ij} + p_{kl} \leq lft_{ij} - est_{kl}$ then either ordering decision is still possible. (case 4)

Whenever a new precedence constraint is posted, say we sequence operation o_{ij} before operation o_{kl}, then est_{kl} and lft_{ij} are updated by

$$est_{kl} = \max\{est_{kl}, est_{ij} + p_{ij}\} \text{ and } \tag{3}$$

$$lft_{ij} = \min\{lft_{ij}, lft_{kl} - p_{kl}\}, \tag{4}$$

and these new values are then propagated forward or backward respectively through all precedence constraints which are pre-specified or previously posted. We call the above process "constraint propagation".

The dominance conditions in CBA, of course, provide only necessary conditions for determining a set of feasible schedules, and, thus, the interleaved application of CBA and constraint propagation yields an incomplete search procedure. What is needed to generate a complete schedule is some heuristics for resolving the undecided states specified by CBA in case 4.

In such situations where CBA leaves the search in a state with several unordered pairs of operations, PCP focuses its attention on the pair whose temporal slacks are currently most constrained. Since the posting of any precedence constraint is only likely to further constrain the temporal slacks of other unordered pairs, delaying the ordering decision for the currently most constrained pair will greatly increases the chances of generating an infeasible schedule.

To estimate the constrainedness for any unordered pair of operations (o_{ij}, o_{kl}), where $m_{ij} = m_{kl}$, PCP first defines the projected temporal slack, if o_{ij} is sequenced before o_{kl}, as

$$Slack(o_{ij} \rightarrow o_{kl}) = lft_{kl} - est_{ij} - (p_{ij} + p_{kl}). \tag{5}$$

Similarly the projected temporal slack, if o_{kl} is sequenced before o_{ij}, can be defined as

$$Slack(o_{kl} \rightarrow o_{ij}) = lft_{ij} - est_{kl} - (p_{ij} + p_{kl}). \tag{6}$$

Intuitively, the unordered pair which is currently most constrained can be defined as the one which has the minimum *slack* among all unordered pairs. However, PCP defines another metric, referred to as biased slack (*Bslack*), in determining the most constrained pair. The motivation behind the creation of *Bslack* is primarily for dealing with the problem in which we can not distinguish the most constrained pair by relying on the minimum *slack* as the only criterion. Instead, for any unordered pair (o_{ij}, o_{kl}), the value of $\max\{Slack(o_{ij} \rightarrow o_{kl}), Slack(o_{kl} \rightarrow o_{ij})\}$ should also be used to bias the value of $\min\{Slack(o_{ij} \rightarrow o_{kl}), Slack(o_{kl} \rightarrow o_{ij})\}$.

Based on $Slack(o_{ij} \rightarrow o_{kl})$ and $Slack(o_{kl} \rightarrow o_{ij})$, the biased slacks for any unordered pair of operations (o_{ij}, o_{kl}), where $m_{ij} = m_{kl}$, are given by the expressions

$$Bslack(o_{ij} \rightarrow o_{kl}) = \frac{Slack(o_{ij} \rightarrow o_{kl})}{\sqrt{S}}, and \tag{7}$$

$$Bslack(o_{kl} \rightarrow o_{ij}) = \frac{Slack(o_{kl} \rightarrow o_{ij})}{\sqrt{S}}, \tag{8}$$

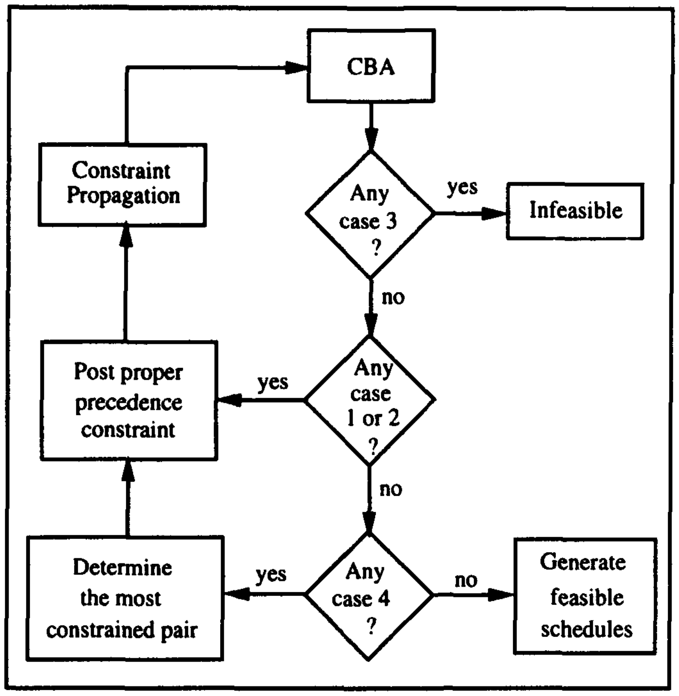

CBA
Constraint Propagation
Any case 3 ?
yes
Infeasible
no
Post proper precedence constraint
yes
Any case 1 or 2 ?
no
Determine the most constrained pair
yes
Any case 4 ?
no
Generate feasible schedules

Figure 9: PCP Algorithm

where

$$S = \frac{min\{Slack(o_{ij} \rightarrow o_{kl}), Slack(o_{kl} \rightarrow o_{ij})\}}{max\{Slack(o_{ij} \rightarrow o_{kl}), Slack(o_{kl} \rightarrow o_{ij})\}} \tag{9}$$

estimates the degree of "similarity" between the value of $Slack(o_{ij} \rightarrow o_{kl})$ and $Slack(o_{kl} \rightarrow o_{ij})$ respectively.

Given the metric of $Bslack$, for any unordered pair of operations (o_{ij}, o_{kl}), where $m_{ij} = m_{kl}$, its degree of constrainedness or 'criticality' is determined by the expression

$$CR(o_{ij}, o_{kl}) = -\min\{Bslack(o_{ij} \rightarrow o_{kl}), Bslack(o_{kl} \rightarrow o_{ij})\} \tag{10}$$

Once PCP has determined the most constrained pair, say (o_{ij}, o_{kl}), which has the maximum $CR(o_{ij}, o_{kl})$, it posts the precedence constraint $(o_{ij} \rightarrow o_{kl})$ if $Bslack(o_{ij} \rightarrow o_{kl}) > Bslack(o_{kl} \rightarrow o_{ij})$. Otherwise, PCP posts the precedence constraint $(o_{kl} \rightarrow o_{ij})$. A tie is broken arbitrarily. Given in Figure 9 is the whole PCP algorithm.

5.3 Job-shop Makespan Scheduling

5.3.1 The proposed method

It is not hard to see that, for any job-shop makespan problem, we can reduce it to a job-shop problem with ready times and deadlines, in which every job has zero ready time and a common deadline which is equal to the minimum makespan of the problem. The objective of the problem is to generate, if there is, a feasible schedule which satisfies the condition that every job is started after its ready time and completed before its deadline. The schedule which minimizes the makespan for the former will be a feasible schedule for the latter. In some sense, there is a dual relationship between the makespan problem and the problem with ready times and deadlines. The common deadline which equals to the minimum makespan is also the least common deadline so that the problem can still find some feasible schedules. We call it the "least feasible common deadline". Based on this relationship, the makespan problem can be viewed as finding the least feasible common deadline for the problem with ready times and deadlines.

If we have a procedure for optimally solving the constraint satisfaction scheduling problem, we could use it within a binary search procedure to obtain an optimal solution for the makespan problem. A natural extension of this would be to obtain an approximation algorithm for the makespan problem by using an approximation algorithm for the constraint satisfaction scheduling problem within the binary search. Thus, we may propose an approximate algorithm by using PCP within a binary search procedure. We start with known upper and lower bounds on the common deadline D, and at each step run PCP for a value of D midway between the current upper and lower bounds. If PCP generates any feasible schedule, D becomes the new lower bound and we continue; if PCP doesn't generate any feasible schedule, D becomes the new upper bound. By the binary search, we can converge to the solution very rapidly.

Unfortunately, binary search will guarantee to give the least feasible common deadline only if PCP holds the monotonicity property. This property states that if for D_1, PCP can not find any feasible schedule, then for any smaller $D_2 (< D_1)$, it should not find any feasible schedule, or vice versa. Since PCP is an approximation algorithm, it normally does not hold the monotonicity property. Without the general monotonicity property, using PCP within the binary search sometimes will not guarantee to deliver the best solution.

Instead, we use a k-iteration search procedure. Within this search procedure, we repeatedly try PCP k times with different common deadlines which are evenly selected between the known upper and lower bounds on D. The main reason of using the k-iteration search procedure is not to solve the non-monotonicity problem in PCP, but to reduce the chances of missing the best solution. The disadvantage of the k-iteration search procedure is the need to determine a proper value for k. With a large k, we can generate better solutions but the computation time becomes higher. In our experience, the k-iteration search performs very well and better than the binary search when k is equal to or greater than 8. Thus, in the final implementation we choose k equal to 8.

In the k-iteration search procedure, the upper bound D_U is determined by running a dispatch procedure with six different priority rules - SPT, LPT, LFT, EFT, MOR, and LOR, and taking the best makespan generated. For generating the lower bound D_L, we apply the procedure previously proposed by Florian, Trepant, and McMahon (1971). This lower bound is obtained by sequencing each machine without regard to the other machines using the earliest start time as the priority rule, and then choosing the maximum completion time among all jobs.

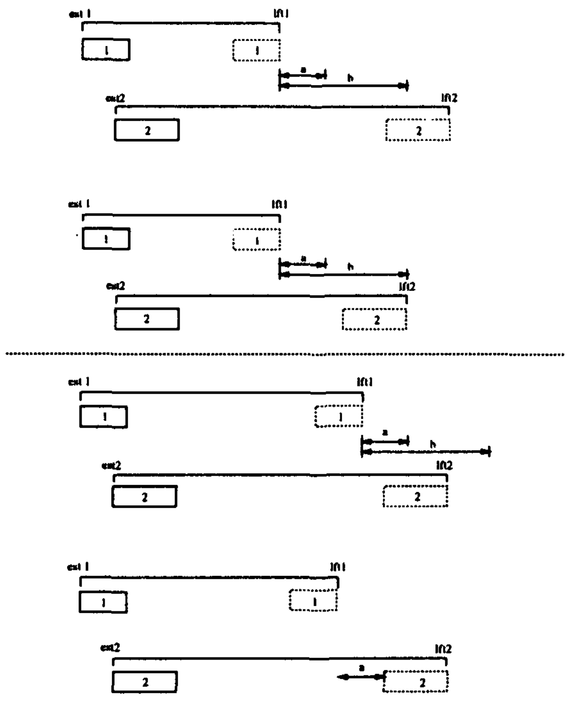


Figure 10: Modified PCP Search Procedure

Notice that for a problem with a common deadline D, if PCP can find a feasible schedule, then the makespan C_{min} is less than or equal to D. Thus, when PCP can find a feasible schedule, C_{min} is what we want, not D. Although our approximation procedure is motivated by searching for the least feasible common deadline for the problem, what we are most interested in is the various values of the makespan associated with the feasible schedules generated during the search. Within the k-iteration search procedure, what we really do is use PCP to generate k schedules for the problem with k different common deadlines evenly selected between the upper and lower bounds. The feasible schedule which has the minimum makespan will be our final solution.

In PCP, the search for feasible schedules is terminated whenever an infeasible state (case 3 in CBA) is reached. In such a situation, rather than just stopping the whole procedure, we might like to continue and generate a complete schedule. The motivation behind this is based on an observation that for certain common deadlines PCP only generates very few number of infeasible states, say one or two. For these common deadlines, if we continue the procedure and construct complete schedules, the values of the final makespan should be very close to the given common deadlines. Therefore, we slightly modify the original PCP procedure, shown in Figure 2, to always generate a complete schedule. Moreover, with the modification new PCP guarantees to always produce better solutions than before, since it includes both feasible and infeasible schedules to search for the best solutions.

Below, we give the description of the algorithm in detail. The algorithm, which is called MULTIPCP, takes as input sets of processing times and routings of jobs and a bound k on the desired number of iterations. Then it goes as follows:

1. Compute upper bound C_U and lower bound C_L.
   Set *Best_Makespan* $= \infty$.
   Set $I = 0$.

2. If $I \geq k$, stop.
   Otherwise, set common deadline $D = C_U - I(C_U - C_L)/k$.

3. Run PCP and compute the makespan M.
   If $M <$ *Best_Makespan*, set *Best_Makespan* $= M$.
   If *Best_Makespan* $= C_L$, stop because we have the optimal solution. Otherwise, set $I = I + 1$. and go to 2.

5.3.2 Performance Study

5.3.3 Generation of test problems

In this section we evaluate the performance of MULTIPCP with one well-known procedure, Shifting Bottleneck (SB) developed by Adams, Balas, and Zawack (1988). Shifting Bottleneck has reported superb performance on job-shop makespan problems in terms of solution quality and computation time used. The implementation of the SB we used in this study was kindly provided to us by Applegate and Cook. For details of this implementation please see Applegate and Cook (1991).

We test both our procedure MULTIPCP and Shifting Bottleneck on two sets of problems previously published in the literature. The first set contains some small problems with size from 6-job and 6-machine to 15- job and 15-machine. The first three problems, Mt06, Mt10, and Mt20, are those notorious problems originally posed by Fisher and Thompson in 1966. The rest of the first set is taken from 40 problems created by Lawrence (1984). Since only 36 problems have reported optimal solutions so far, we just included those problems in the set.

The second set of the problems were generated by Taillard (1993). This set contains several large job-shop makespan problems with size from 15-job and 15-machine to 50-job and 20-machine. One random number generator and several procedures for generating problem data were also provided by Taillard. For each problem, Taillard gave both time seed and machine seed to generate the required random numbers and he also reported the best solution obtained through a Taboo search using very extensive computational resource. The problems reported by Taillard were the most difficult ones among several hundreds which were also generated and solved by him. Thus, these problems should be described as hard problems.

To measure the quality of the solutions generated by both MULTIPCP and SB, we calculate percentage deviation from the optimal solution for the first set of problems and percentage deviation from the best solution for the second set respectively. In addition to the quality of the solutions, we also report how many CPU seconds are used by each procedure to solve each problem. To achieve a fair comparison for the computational performance, both

MULTIPCP and SB were written in C and implemented on a Sun IPX. For MULTIPCP, the bound k on the number of iterations was chosen to be 8. The experimental results on the first set of problems appear in Table 2. In Table 3, we give the average performance respectively for SB and MULTIPCP, and also for two new versions of Shifting Bottleneck (SBIII and SBIV), recently developed by Balas, Lenstra, and Vazacopoulos (1993), on the set of small problems. For SBIII and SBIV, the values for the percentage deviation from the optimal solution and the CPU second are computed based on the results reported by Balas et al. (1993), and we fairly adjust the running times for SBIII and SBIV, since they correspond to a SUN SPARC330. Then, in Table 4 and 5 we give the results for MULTIPCP and SB on the second set of large problems. Finally, the average performance for SB and MULTIPCP on the set of large problems is shown in Table 6.

5.3.4 Discussion

The results indicate that, on average, MULTIPCP performs slightly better than SB for both the small and large problems. For the set of small problems, the average percentage deviation from the optimal solution generated by SB is 3.0% versus 1.7% by MULTIPCP. Moreover, for this set of small problems, MULTIPCP can also generate very comparable results with two new versions of Shifting Bottleneck, SBIII and SBIV. These results indicate that for small problems MULTIPCP can produce near-optimal solutions. For the set of large problems, both SB and MULTIPCP moderately deteriorate their performance. The average percentage deviation from the best solution by SBI is 9.31% versus 8.34% by MULTIPCP. On this set of problems, MULTIPCP still generates very competitive performance to the SB procedure.

For the computational performance, MULTIPCP needs a little more CPU time than SB on the set of small problems, but the difference is just a second. However, when running SB and MULTIPCP on the set of large problems, we realize that MULTIPCP uses much less CPU time than SB. To illustrate, on the problems with the largest size (50 jobs and 20 machines), the average CPU time used by MULTIPCP is 242.64 seconds versus 414.08 seconds by SB.

By no means, we can conclude that MULTIPCP is better than SB or vice versa. From the testing results we can see that SB performs better than MULTIPCP on the problems in which we have high ratio of number of jobs to number of machines such as the problems of 30-job and 10-machine and also those of 50-job and 20-machine. Except for these problems, MULTIPCP has better performance than SB. Based on these observations, we recommend that in a situation in which the problems has high job-machine ratio, SB could be used. Otherwise, MULTIPCP could be used for the problems in which the job-machine ratio is low or the problem size is so large that the computational cost becomes more important.

Table 2: % from optimal solution and computation time for SB and MULTIPCP on small problems

	Job x	Optimal	SB			MULTIPCP		
Problem	Machine	Value	Value	%	CPU sec	Value	%	CPU sec
mt06	6 x 6	55	59	7.273	0.12	55	0	0.13
mt10	10 x 10	930	952	2.366	1.08	949	2.043	1.10
mt20	20 x 5	1165	1228	5.408	0.63	1192	2.318	4.22
la1		666	666	0	0.10	666	0	0.10
la2		655	684	4.427	0.13	670	2.290	0.48
la3	10 x 5	597	605	1.340	0.12	617	3.350	0.47
la4		590	603	2.203	0.18	600	1.695	0.47
la5		593	593	0	0.12	593	0	0.03
la6		926	926	0	0.20	926	0	0.05
la7		890	890	0	0.23	890	0	1.52
la8	15 x 5	863	863	0	0.25	878	1.738	1.42
la9		951	951	0	0.17	951	0	0.05
la10		958	958	0	0.20	958	0	0.05
la11		1222	1222	0	0.25	1222	0	0.05
la12		1039	1039	0	0.18	1039	0	0.05
la13	20 x 5	1150	1150	0	0.25	1150	0	0.05
la14		1292	1292	0	0.22	1292	0	0.07
la15		1207	1207	0	0.25	1207	0	0.50
la16		945	1076	13.862	0.62	982	3.915	0.73
la17		784	829	5.740	0.85	787	0.383	0.63
la18	10 x 10	848	855	0.824	1.00	886	4.481	0.65
la19		842	863	2.494	1.03	852	1.188	0.62
la20		902	918	1.774	1.05	922	2.217	0.62
la22		927	968	4.423	1.87	965	4.099	2.82
la23	15 x 10	1032	1071	3.779	2.48	1041	0.872	2.72
la24		935	1015	8.556	2.07	983	5.134	2.42
la25		977	1061	8.598	1.60	1026	5.015	2.63
la26		1218	1393	14.368	2.27	1272	4.433	7.05
la28	20 x 10	1216	1281	5.345	2.98	1275	4.852	6.58
la30		1355	1355	0	3.45	1356	0.074	7.32
la31		1784	1784	0	7.42	1784	0	15.92
la32		1850	1850	0	4.73	1850	0	0.15
la33	30 x 10	1719	1719	0	3.63	1722	0.175	23.65
la34		1721	1721	0	5.28	1744	1.336	24.77
la35		1888	1888	0	4.80	1888	0	3.26
la36		1268	1326	4.574	4.93	1321	4.180	2.97
la37	15 x 15	1397	1471	5.297	3.25	1446	3.508	3.57
la39		1233	1301	5.515	6.45	1286	4.298	3.92
la40		1233	1347	9.246	6.48	1272	3.163	3.72
Average				3.011			1.712	

Table 3: Average % from optimal solution and computation time for SB, MULTIPCP, SBIII, and SBIV on small problems

	SB	MULTIPCP	SBIII	SBIV
%	3.0	1.7	1.5	1.3
CPU sec	1.9	3.2	4.1	6.6

Table 4: % from best solution and computation time for SB and MULTIPC on large problems

Job x	Best	SB			MULTIPCP		
Machine	Value	Value	%	CPU sec	Value	%	CPU sec
	1231	1360	10.48	6.28	1280	3.98	3.25
	1252	1367	9.19	9.25	1308	4.47	3.60
	1223	1352	10.55	9.63	1288	5.31	3.53
	1181	1289	9.15	6.73	1253	6.10	3.40
15 x 15	1234	1359	10.13	4.92	1306	5.83	3.72
	1243	1314	5.71	7.42	1320	6.19	3.55
	1228	1322	7.65	5.07	1307	6.43	3.18
	1221	1345	10.16	8.57	1289	5.57	3.52
	1289	1437	11.48	8.10	1366	5.97	3.85
	1261	1331	5.55	6.83	1334	5.79	3.50
	1376	1481	7.63	15.10	1488	8.14	10.13
	1381	1503	8.83	17.23	1495	8.25	8.37
	1367	1521	11.27	12.37	1478	8.12	8.97
	1355	1540	13.65	10.83	1452	7.16	8.72
20 x 15	1366	1532	12.15	17.60	1486	8.78	10.22
	1371	1511	10.21	12.90	1478	9.99	9.05
	1480	1605	8.45	13.93	1609	8.72	9.85
	1432	1532	6.98	14.52	1503	4.96	10.80
	1361	1504	10.51	16.35	1440	5.80	9.63
	1373	1535	11.80	11.87	1441	4.95	8.73
	1663	1814	9.08	22.22	1764	6.07	10.20
	1626	1776	9.23	35.65	1739	6.95	11.17
	1574	1725	9.59	29.65	1687	7.18	10.55
	1660	1822	9.76	25.73	1773	6.81	10.33
20 x 20	1598	1847	15.58	24.78	1742	9.01	11.07
	1679	1861	10.84	25.17	1759	4.76	10.87
	1704	1857	8.98	29.00	1862	9.27	11.62
	1626	1809	11.25	34.02	1749	7.56	10.97
	1635	1771	8.32	40.43	1761	7.71	9.40
	1614	1729	7.13	38.77	1743	7.99	11.27

Table 5: % from best solution and computation time for SB and MULTIPC on large problems (continued)

Job x	Best	SBI			MULTIPCP		
Machine	Value	Value	%(PDB)	CPU sec	Value	%(PDB)	CPU sec
	1770	2017	13.95	27.32	2001	13.05	36.45
	1853	2061	11.23	38.83	2019	8.96	34.75
	1855	1987	7.12	33.55	2053	10.67	32.00
	1851	1996	7.83	28.67	2071	11.89	34.63
30 x 15	2007	2140	6.63	32.18	2106	4.93	32.35
	1844	1965	6.56	36.65	2016	9.33	36.27
	1822	1976	8.45	36.25	1954	7.25	32.13
	1714	1915	11.73	40.07	1853	8.11	34.70
	1824	1890	3.62	32.92	2010	10.20	35.88
	1723	1838	6.67	33.43	1931	12.07	33.57
	2064	2343	13.52	67.32	2323	12.55	41.05
	1983	2199	10.89	77.68	2205	11.20	38.90
	1896	2123	11.97	79.28	2165	14.20	42.97
	2031	2392	17.77	62.58	2252	10.88	44.37
30 x 20	2032	2262	11.32	72.83	2215	9.01	41.53
	2057	2329	13.22	80.13	2269	10.31	45.88
	1947	2202	13.10	80.25	2104	8.06	41.62
	2005	2191	9.28	67.38	2277	13.57	43.52
	2013	2270	12.77	60.88	2330	15.75	45.12
	1973	2301	16.62	57.15	2188	10.90	40.35
	2921	3098	6.06	337.05	3177	8.76	247.83
	3002	3237	7.83	404.93	3244	8.06	244.87
	2835	2976	4.97	762.60	3047	7.48	238.00
	2775	2879	3.75	259.63	2951	6.34	234.97
50 x 20	2800	2978	6.36	663.48	3061	9.32	245.95
	2914	3089	6.01	241.07	3196	9.68	234.37
	2895	3105	7.25	248.12	3165	9.33	244.10
	2835	2918	2.93	408.33	3048	7.51	235.37
	3097	unkn	unkn	unkn	3419	10.40	255.62
	3075	3163	2.80	401.53	3349	8.91	245.27

Table 6: Average % from best solution and computation time for SB and MULTIPCP on large problems

Job x Machine	SB		MULTIPCP	
	%	CPU sec	%	CPU sec
15 x 15	9.01	7.28	5.56	3.51
20 x 15	10.15	14.27	7.49	9.45
20 x 20	9.98	30.54	7.33	10.75
30 x 15	8.38	33.99	9.65	34.27
30 x 20	13.05	70.55	11.64	42.53
50 x 20	5.33	414.08	8.38	242.64
Overall	9.31		8.34	

5.4 Job-Shop Weighted-Tardiness Scheduling

To adapt PCP to the weighted-tardiness problem, two basic issues must be addressed. The first issue concerns the inappropriateness of using temporal slack as the only basis for estimating the criticality of various unordered pairs. Second, since CBA depends on the assumption that deadlines are non-relaxable, its advantage as a search-space pruning mechanism is completely lost for the weighted-tardiness problem.

5.4.1 The metric for measuring criticality

To address the first issue, along with the temporal slack, we define another metric, which is referred to as *Tard*, to serve the purpose of estimating the criticality of unordered pair of operations. This metric is defined by computing the increase in tardiness cost resulting from alternative ordering decisions for a given pair of operations. For any unordered pair of operations (o_{ij}, o_{kl}) the 'projected job tardiness after sequencing o_{ij} before o_{kl}', $Tard(o_{ij} \rightarrow o_{kl})$, can be obtained by the following procedure:

1. Set $est_{kl} = \max\{est_{kl}, est_{ij} + p_{ij}\}$.
   Propagate est_{kl} forward through all
   precedence constraints which are pre-specified
   and previously posted on the current search space.
2. Compute the increase in job tardiness.
3. Undo step 1.

The projected job tardiness $Tard(o_{kl} \rightarrow o_{ij})$, if we sequence o_{kl} before o_{kl}, can be computed by the similar manner.

In case that both $Tard(o_{ij} \rightarrow o_{kl})$ and $Tard(o_{kl} \rightarrow o_{ij})$ equal to zero, we defines the projected temporal slack, if we sequence o_{ij} before o_{kl}, as

$$Slack(o_{ij} \rightarrow o_{kl}) = lft_{kl} - est_{ij} - (p_{ij} + p_{kl}), \tag{11}$$

and similarly the projected temporal slack, if we sequence o_{kl} before o_{ij}, as

$$Slack(o_{kl} \rightarrow o_{ij}) = lft_{ij} - est_{kl} - (p_{ij} + p_{kl}). \tag{12}$$

Given *Tard* and *Slack*, for any unordered pair of operations (o_{ij}, o_{kl}), where $m_{ij} = m_{kl}$, its criticality is estimated by the expression

$$CR(o_{ij}, o_{kl}) = Tard(o_{ij} \rightarrow o_{kl}) + Tard(o_{kl} \rightarrow o_{ij}), \tag{13}$$

or

$$CR(o_{ij}, o_{kl}) = -(Slack(o_{ij} \rightarrow o_{kl}) + Slack(o_{kl} \rightarrow o_{ij})), \tag{14}$$

if all unordered pairs have zero look-ahead job tardiness, i.e. no precedence constraint posting will increase the job tardiness.

With *CR*, we can propose a general procedure, referred to as G1, as follows:

Step 1. Initialization.

1. Initialize the earliest start time est_{ij} and the latest finish time lft_{ij} for every operation o_{ij}.
2. For any pair, compute *Tard*, *Slack* and *CR*.

Step 2. If all pairs are ordered, stop. Otherwise,

Step 3. Pick the unordered pair whose *CR* value is maximum.

Step 4. Post the proper precedence constraint.

Step 5. Propagate the earliest start and latest finish times.

Step 6. For remaining pairs, update *Tard*, *Slack*, and *CR*. Go to Step 2.

In Step 4, once we have determined the most critical pair, say (o_{ij}, o_{kl}), which has the maximum *CR* value, we post the precedence constraint $(o_{ij} \rightarrow o_{kl})$ if $Tard(o_{ij} \rightarrow o_{kl}) < Tard(o_{kl} \rightarrow o_{ij})$, or if $Slack(o_{ij} \rightarrow o_{kl}) > Slack(o_{kl} \rightarrow o_{ij})$ in case that $Tard(o_{ij} \rightarrow o_{kj}) = Tard(o_{kl} \rightarrow o_{ij}) = 0$. Otherwise, we post the precedence constraint $(o_{kl} \rightarrow o_{ij})$. A tie is broken arbitrarily.

One major reason why PCP performs so well for the problem with deadlines is because it incorporates a powerful mechanism to prune infeasible regions on the search space, a mechanism provided by *Constraint-Based Analysis* (CBA). Since CBA depends on the assumption that deadlines are non-relaxable, it provides no leverage it in the weighted-tardiness problem. In the next section, we'll propose an alternative mechanism, which uses a dispatching rule to guide the proper posting of precedence constraints.

5.4.2 Using a dispatching rule to guide posting

Image that after we determine the unordered pair which has the maximum CR value, we use a dispatching rule to generate two schedules, one that assumes each of the possible precedence constraints associated with these two operations. For example, suppose that the unordered pair (o_{ij}, o_{kl}) is the one whose CR value is maximum. Then we generate two dispatching schedules for the original problem with the precedence constraints posted so far and also the precedence constraints $(o_{ij} \rightarrow o_{kl})$ and $(o_{kl} \rightarrow o_{ij})$ respectively. If the schedule associated with the precedence constraint $(o_{ij} \rightarrow o_{kl})$ is better than the other one, then we post $(o_{ij} \rightarrow o_{kl})$. Otherwise, we post the precedence constraint $(o_{kl} \rightarrow o_{ij})$.

Now, if we also keep the best schedule after we post the proper precedence constraint, then, when we go on to the next step, we only need to generate one dispatch schedule. Let's take an example. Suppose that the unordered pair (o_{ij}, o_{kl}) has now the maximum CR value. Then, in the current best schedule, either o_{ij} is sequenced before o_{kl} or o_{kl} is sequenced before o_{ij}. If o_{ij} is sequenced before o_{kl} in the current best schedule, then we generate a dispatch schedule for the precedence constraint $(o_{kl} \rightarrow o_{ij})$. Otherwise, we generate a dispatch schedule for the precedence constraint $(o_{ij} \rightarrow o_{kl})$. Therefore, we only need to generate one dispatch schedule for the precedence constraint which is not in the current best schedule. After we generate the new schedule, we compare it with the current best schedule. If the new one is better, we retain it and post the precedence constraint associated with it. Otherwise, we post the other precedence constraint and discard the new schedule.

We can further reduce the computational cost by introducing a heuristic assumption that distinguishes between 'easy' and 'hard' ordering decisions. To explain this scheme, suppose that we have selected the most critical pair. Then, we first use Step 4 in the general procedure **G1** to decide which precedence constraint should be posted. If the precedence constraint suggested by Step 4 is already in the current best schedule, then we can skip the generation of a new dispatch schedule and continue to the next step. With this scheme, we generate a new dispatch schedule only when the precedence constraint suggested by Step 4 is not in the current best schedule. The supposition underlying this scheme is based on the observation that examining all ordering decisions that must be made during the overall schedule-generation process, we can identify some 'easy' and 'hard' ones. The 'easy' decisions are those that different methods or priority rules will give the same answers. On the other hand, the 'hard' decisions are those that are difficult to make such that different methods or rules will give different answers. If we assume that the answers for easy decisions correspond to those precedence constraints which are suggested by Step 4 and are also in the current best schedule, then we can accept these answers without generating any new schedule.

Now, we present the final procedure, referred to as Weighted-Tardiness Version of Precedence Constraint Posting (WTPCP). Given the problem data of processing times, job routings, job release dates, job due dates and job weights, WTPCP proceeds as following:

Step 1. Initialize:

- Set $\overline{S_1} = \phi$ and $\overline{S_2} = \phi$.
- Generate a dispatch schedule by a dispatching rule.

```
• For each operation $o_{ij}$, compute the earliest start time $est_{ij}$
  and the latest finish time $lft_{ij}$.
• $\forall(o_{ij}, o_{kl}) \in S$, where $S$ is the set of all pair of operations,
  compute $Tard(o_{ij} \rightarrow o_{kl})$ and $Tard(o_{kl} \rightarrow o_{ij})$,
  IF $Tard(o_{ij} \rightarrow o_{kl}) > 0$ or $Tard(o_{kl} \rightarrow o_{ij}) > 0$ THEN add $(o_{ij}, o_{kl})$ to $\overline{S_1}$
  ELSE compute $Slack(o_{ij} \rightarrow o_{kl})$, and $Slack(o_{kl} \rightarrow o_{ij})$ and add $(o_{ij}, o_{kl})$ to $\overline{S_2}$.
Step 2. If $\overline{S1} = \phi$ and $\overline{S2} = \phi$, stop.
Step 3. Choose the pair with the highest criticality:
    IF $\overline{S_1} \neq \phi$
        THEN $\forall(o_{ij}, o_{kl}) \in \overline{S_1}$, choose the pair $(o^*_{ij}, o^*_{kl})$,
            such that $CR(o^*_{ij}, o^*_{kl})$ is maximum. Break ties arbitrarily.
        ELSE $\forall(o_{ij}, o_{kl}) \in \overline{S_2}$, choose the pair $(o^*_{ij}. o^*_{kl})$,
            such that $CR(o^*_{ij}, o^*_{kl})$ is maximum. Break ties arbitrarily.

Step 4. Post the proper precedence constraint:
    IF $(o^*_{ij}, o^*_{kl}) \in \overline{S_1}$
        IF $Tard(o^*_{ij} \rightarrow o^*_{kl}) < Tard(o^*_{kl} \rightarrow o^*_{ij})$,
            check the precedence constraint $(o^*_{ij} \rightarrow o^*_{kl})$.
        ELSE check the precedence constraint $(o^*_{kl} \rightarrow o^*_{ij})$ .
        Delete $(o^*_{ij}, o^*_{kl})$ from $\overline{S_1}$.
    IF $(o^*_{ij}, o^*_{kl}) \in \overline{S_2}$
        IF $Slack(o^*_{ij} \rightarrow o^*_{kl}) > Slack(o^*_{kl} \rightarrow o^*_{ij})$
            check the precedence constraint $(o^*_{ij} \rightarrow o^*_{kl})$.
        ELSE check the precedence constraint $(o^*_{kl} \rightarrow o^*_{ij})$ .
        Delete $(o^*_{ij}, o^*_{kl})$ from $\overline{S_2}$.

Step 5. Propagate earliest start and latest finish times.
Step 6. Update Tard and Slack:
    $\forall(o_{ij}, o_{kl}) \in \overline{S_1}$ update $Tard(o_{ij} \rightarrow o_{kl})$ and $Tard(o_{kl} \rightarrow o_{ij})$.
    $\forall(o_{ij}, o_{kl}) \in \overline{S_2}$ compute $Tard(o_{ij} \rightarrow o_{kl})$ and $Tard(o_{kl} \rightarrow o_{ij})$.
        IF $Tard(o_{ij} \rightarrow o_{kl}) > 0$ or $Tard(o_{kl} \rightarrow o_{ij}) > 0$
            THEN delete $(o_{ij}, o_{kl})$ from $\overline{S_2}$ and add to $\overline{S_1}$.
            ELSE update $Slack(o_{ij} \rightarrow o_{kl})$, and $Slack(o_{kl} \rightarrow o_{ij})$.
    Go to step 2.
```

And the procedure to check any given precedence constraint $(o_{ij} \rightarrow o_{kl})$ is given by:

```
Check $(o_{ij} \rightarrow o_{kl})$:
    IF $(o_{ij} \rightarrow o_{kl})$ is in the current schedule, post $(o_{ij} \rightarrow o_{kl})$ .
    ELSE Generate a new dispatch schedule with the precedence
            constraints posted so far and $(o_{ij} \rightarrow o_{kl})$.
            IF the new schedule is better, retain it and post $(o_{ij} \rightarrow o_{kl})$ .
            ELSE discard the new schedule and post $(o_{kl} \rightarrow o_{ij})$ .
```

5.4.3 WTPCP as a schedule improvement procedure

Although the original motive behind the use of a dispatching rule is to guide the general procedure **G1** in properly posting precedence constraints, from the view point of the rule,

G1 acts like a knowledge source, which continuously provides valuable information about where we should interchange pair of operations, generate a new neighborhood, and search for the better schedules. Whenever **G1** suggests for a pair of operations a precedence constraint which is not consistent with the current schedule, we interchange this pair and generate a new schedule. If the new schedule is better, we retain it; otherwise, we keep the current schedule. Based on this description, WTPCP can also be characterized as a 'schedule improvement procedure, which continuously improve the solutions generated by the simple dispatching rules.

One attribute of WTPCP is that we don't have to randomly interchange any pair of operations when searching for better schedules. Instead, we limit the interchanges considered to those seemingly promising pairs, based on one-step look-ahead information about the projected increase in tardiness cost.

5.4.4 Computational Study

5.4.5 Data generation

In this set of experiments we randomly generate data for a 10-machine job shop. Jobs arrive continuously with inter-arrival times taken from an exponential distribution, which means that the number of jobs arriving within an interval of given length is Poisson-distributed. The arrival rate is determined by the queueing expression, $\lambda = \overline{U}m\mu$, where λ is the arrival rate, $\overline{U}$ the utilization of the shop, m the number of machines, and μ the average processing rate. Each job has 1 to 10 operations, and no two consecutive operations require the same machine. The actual number of operations is a random variable, drawn from the uniform distribution $U[1,10]$. Totally, we generate 200 jobs. The process routings are randomly generated such that each machine is equally likely to perform a job's next operation. The processing times are from a uniform distribution, $p_i \sim U[1,30]$. Job weights are also drawn from the same uniform distribution, $w_j \sim U[1,30]$.

In the study by Kanet and Hayya (1982), the average utilization of actual shops were found between 80% and 90%. Thus, in these experiments the arrival rates are generated in such a manner to yield two approximate utilization levels of 80% and 90%. Due dates of the arriving jobs are set by the total work content (TWK) rule suggested by Baker (1984), such that the flow allowance of a job is proportional to the total work of that job, i.e., $d_i = r_i + \alpha \sum_{k=1}^{n_i} p_{ik}$, where α is the flow allowance factor. In order to examine the performance of WTPCP across a range of shop conditions, we choose three flow allowance factors, which are 2, 3 and 4, to generate different scenarios with vary tight due-date, loose due-date, and very loose due-date jobs. For each combination of shop utilization and due-date tightness, we generate 10 random problems and the same set of job arrival times, job routings, job weights, and processing times is used for each dispatching rule. In total, therefore, 60 problems are solved by the integrated method of WTPCP with different rules respectively.

Our main concern in this study is to investigate the effectiveness of WTPCP in improving the solutions generated by dispatching rules. Mean weighted-tardiness is our major criterion.

Two other criteria are also investigated in this study. They are mean weighted flowtime and proportion of tardy jobs.

5.4.6 Dispatching rules

We select some best-known dispatching rules in this computational study. Some of them were originally developed for unweighted-tardiness problems and others were for weighted tardiness. The rules are selected primarily based on their popularity and reported good performance in the earlier literature of job-shop scheduling.

WSPT represents a simple, popular, widely used rule in most prior scheduling research and in real-world applications. WSPT has been confirmed by many studies to be effective at reducing the mean weighted tardiness when shop utilization is high and job due dates are very tight.

Since Carroll first proposed COVERT (Cost Over Time) in 1965, many researchers have extensively investigated and tested this rule on problems under different shop conditions. Even though there have been many new rules developed since then, COVERT is still producing very effective and robust performance in minimizing mean tardiness as observed by Blackstone et al. (1982) and Russell et al. (1987). The modified rule, called WCOVERT, that we used in the study is a weighted version of COVERT extended by taking job weights into account. This modification on COVERT was suggested by Vepsalainen and Morton (1987). For setting the best values for the slack parameter b and the look-ahead parameter k used in WCOVERT, we run a preliminary parameter tuning test and choose $b = 2$ and $k = 2$ to be used in the subsequent experiments.

Recently, Baker and Kanet (1983) developed a dispatching rule known as the modified operation due date (MOD) rule, which has been shown to provide outstanding performance in minimizing mean tardiness, as reported by Baker (1984). WMOD is a weighted version of MOD, which has the form, w_j/mod_{ij}, where w_j is the weight of job j and mod_{ij} is the modified operation due date of operation i in job j. For setting the operational due-dates or milestones, we use the TWK rule which normally gives the very best performance, as suggested by Baker and Kanet (1983).

Closely related to the modified operation due date (MOD) rule, Anderson and Nyirenda (1990) developed two new rules which are the extended versions of MOD, by introducing the notion of dynamic operation due date. The new rule are called CR+SPT and S/RPT+SPT. CR+SPT is the rule which combines the shortest processing time (SPT) rule and critical ratio (CR) rule in a natural way, and S/RPT+SPT is the rule which combines the slack per remaining work (S/RPT) rule and the shortest processing time (SPT) rule. The basic idea behind the development of these rules is that every time after we dispatch one operation of a job, we need to modify the operation due dates for the remaining operations of the same job, because the remaining flow allowance within the job has been changed. Hence, the operation due dates should be dynamic rather than static. Like MOD rule, rules CR+SPT and S/RPT+SPT do not require any parameter estimation and they have been shown superior

Table 7: Mean weighted tardiness (MWT) and % improvement

		Utilization = 80%			Utilization = 90%		
	Priority	initial	final		initial	final	
α	Rule	MWT	MWT	% impv.	MWT	MWT	% impv.
2	WSPT	268.62	229.53	14.55	317.96	276.74	12.96
	WCOVERT	227.37	185.88	18.25	268.04	230.68	13.94
	ATC	241.90	209.92	13.22	302.65	265.23	12.36
	WMOD	205.21	177.21	13.69	263.42	231.59	12.08
	CR+SPT	205.30	180.99	11.84	259.96	229.64	11.66
	S/RPT+SPT	234.51	201.46	14.10	290.75	251.20	13.60
3	WSPT	85.71	60.70	29.18	110.41	86.63	21.54
	WCOVERT	36.53	24.04	34.19	58.67	44.56	24.04
	ATC	29.38	22.33	23.98	53.58	42.03	21.56
	WMOD	31.56	21.26	32.63	53.56	41.71	22.12
	CR+SPT	35.97	27.12	24.60	46.54	35.87	22.93
	S/RPT+SPT	44.38	30.94	30.29	58.35	41.90	28.19
4	WSPT	32.39	23.17	28.47	93.70	73.41	21.65
	WCOVERT	3.80	1.20	68.33	30.32	22.41	26.09
	ATC	6.21	3.37	45.73	33.57	25.73	23.37
	WMOD	5.86	3.15	46.31	41.17	29.71	27.84
	CR+SPT	8.83	6.56	25.72	31.78	24.97	21.41
	S/RPT+SPT	12.73	7.82	38.63	35.50	27.84	21.57

performance in comparison with rules MOD and COVERT.

The development of the rule ATC has evolved from the R&M rule developed by Rachamadugu and Morton (1981) for the single-machine tardiness problem. Vepsalainen and Morton (1987) modified the R&M rule for the job-shop weighted-tardiness problem and called the apparent tardiness cost (ATC) rule. ATC was reported to outperform EDD, S/RPT, WSPT, and COVERT for mean weighted tardiness and proportion of tardy jobs. Since ACT is a parameterized rule, the values of parameters b and k are very sensitive to its overall performance. To set the best values for these parameters, we run a preliminary parameter tuning test and choose $b = 1$ and $k = 2$ to be used in the subsequent experiments.

5.4.7 Results and discussion

Table 7 summarize results with respect to mean weighted tardiness (MWT) performance at different levels of due-date tightness across a range of shop utilization. For each dispatching rule we first give the initial solution in terms of MWT and then the final improved one generated by our new procedure, WTPCP. These values of MWT are based on averages over ten sample replications for each problem setting. To demonstrate the amount of improvement produced by WTPCP, we compute the percentage of improvement (% impv.) by the formula, $\frac{MWT_{initial}-MWT_{final}}{MWT_{initial}} \times 100$, for each dispatching rule.

Results in Table 7 indicate that significant improvement over local dispatching rules can be achieved through the coordination of the global precedence-constraint information. Examining the results shown in Table 7, we can realize that WTPCP has performed remarkably well at reducing the mean weighted tardiness of the schedules generated by dispatching rules. The typical percentage of the improvement for a particular rule ranges from 10% to 30%. In one extreme case WTPCP can even produce 68.33% improvement for rule WCOVERT in the group of problems with shop utilization 80% and flow allowance factor 4. For the group of problems with very loose due dates and very low shop utilization, we observe the greatest improvement that is from 26% for rule CR+SPT to 68% for rule WCOVERT. Then the amount of improvement decreases when job due dates becomes more tight and the shop becomes more congested. This phenomena may imply that most of the dispatching rules can perform quite well on the problems with very tight due dates and very high shop utilization, in which dispatching rules leave very little room to WTPCP for further improvement. Even so, WTPCP can still generate average improvement from 12% for rule S/RPT+SPT to 14% for rule CR+SPT in the group with the tightest due dates and the highest shop utilization. Finally, with respect to the concerns of computational cost, WTPCP implemented in C took about 80 seconds to solve each problem on a SUN IPX machine.

In applying the dispatching rules to real applications, people have often experienced the "crossover" phenomenon identified by Baker (1984). This phenomenon describes the situations that some rules performs very well in shops with low utilization and loose due dates but may deteriorate in congested ones with tight due dates, or vice versa. Because of this crossover phenomenon, there does not exist a universally-accepted rule which can dominate others for any performance criterion under different shop conditions. In order to reduce the impact of this phenomenon, we can apply several dispatching rules simultaneously and choose the best one. We call this the 'best-rule' strategy.

Based on the best-rule strategy, in Table 8 we present the relative performance of WTPCP to the best rule. The relative performance is computed by the expression, $\frac{MWT_{WTPCP}}{MWT_{best\_rule}}$, where MWT_{best_rule} is the mean weighted tardiness generated by the best rule and MWT_{WTPCP} the improved value by WTPCP. We also give the relative performance of each dispatching rule to the best rule using the same expression. Finally, for demonstrating the overall performance of WTPCP and each rule with respect to the best rule, on the bottom of Table 8 we compute the average relative performance over a range of shop utilization for each group of problems with different due-date tightness.

As Table 8 indicates, based on the best-rule strategy WTPCP can improve the MWT performance of the best rule, on average, by about 12% when due dates are very tight, 24% when due dates are loose, and 43% when due dates are very loose.

Although the main objective of these computational experiments is to study the performance improvement produced by WTPCP, some insights can be provided on the MWT performance

Table 8: Relative performance of mean weighted tardiness with respect to the best rule

		Flow Allowance Factor		
		$\alpha = 2$	$\alpha = 3$	$\alpha = 4$
Shop Utilization	Priority Rule	Relative Performance	Relative Performance	Relative Performance
80%	WTPCP	0.883	0.750	0.345
	WSPT	1.394	3.277	10.482
	WCOVERT	1.180	1.397	1.230
	ATC	1.255	1.124	2.010
	WMOD	1.065	1.207	1.896
	CR+SPT	1.065	1.376	2.858
	S/RPT+SPT	1.217	1.697	4.106
90%	WTPCP	0.880	0.776	0.803
	WSPT	1.289	2.590	3.686
	WCOVERT	1.095	1.376	1.193
	ATC	1.236	1.257	1.321
	WMOD	1.076	1.256	1.620
	CR+SPT	1.062	1.092	1.250
	S/RPT+SPT	1.188	1.369	1.397
Average	WTPCP	0.882	0.763	0.574
	WSPT	1.342	3.549	7.084
	WCOVERT	1.138	1.387	1.212
	ATC	1.246	1.191	1.665
	WMOD	1.071	1.232	1.758
	CR+SPT	1.064	1.234	2.054
	S/RPT+SPT	1.203	1.533	2.752

Table 9: Percentage of tardy jobs (PT)

		Utilization = 80%		Utilization = 90%	
	Priority	initial	final	initial	final
α	Rule	PT	PT	PT	PT
	WSPT	38.6	36.5	44.3	41.0
	WCOVERT	39.2	35.2	45.3	41.3
	ATC	46.5	42.0	55.0	51.8
2	WMOD	39.2	35.2	46.0	42.7
	CR+SPT	40.2	36.8	46.9	44.2
	S/RPT+SPT	38.1	35.5	42.8	41.5
	WSPT	16.7	13.8	18.6	16.1
	WCOVERT	12.6	8.7	15.8	12.0
	ATC	17.0	14.3	21.2	18.2
3	WMOD	11.9	9.2	14.3	10.6
	CR+SPT	12.5	9.8	15.8	12.5
	S/RPT+SPT	12.0	9.3	13.8	10.7
	WSPT	7.9	5.8	12.7	10.4
	WCOVERT	2.8	0.9	7.2	4.8
	ATC	5.2	3.7	12.2	10.1
4	WMOD	3.8	2.3	7.7	5.3
	CR+SPT	5.6	4.4	8.9	6.6
	S/RPT+SPT	5.1	3.5	7.8	6.5

of the selected rules. In this set of experiments we do not find an overall winer among all rules with respect to the MWT criterion. It seems that rules WMOD and CR+SPT perform quite well on the group of problems with very tight due dates. However, rules WCOVERT and ATC perform much better when the due dates are very loose. With regard to the rule S/RPT+SPT, we only find out that it is a mediocre performer on the MWT criterion and it performs relatively poor when the due dates are very loose. As to WSPT rule, it finishes last in all three groups of problems.

Finally Tables 9 and 10 refer to the results with respect to the criteria of percentage of tardy jobs (PT) and mean weighted flowtime (MWFT). We can see from Table 9 that WTPCP is also very effective at reducing the percentage of tardy jobs at all levels of due-date tightness under different shop utilization. However, for the MWFT criterion WTPCP sometimes produces larger values than the dispatching rules when job due dates become very loose and shop utilization still remains very low, which can be seen in Table 10. It seems that in order for WTPCP to reduce the mean weighted tardiness some idleness has been inserted in the schedule so that the final job flowtimes become increased.

Table 10: Mean weighted flowtime (MWFT)

		Utilization = 80%		Utilization = 90%	
α	Priority Rule	initial MWFT	final MWFT	initial MWFT	final MWFT
2	WSPT	2316.42	2209.30	2274.11	2177.30
	WCOVERT	2367.73	2228.84	2337.36	2216.07
	ATC	2456.20	2304.62	2434.72	2282.50
	WMOD	2399.56	2251.35	2394.73	2250.95
	CR+SPT	2337.35	2229.93	2329.40	2203.72
	S/RPT+SPT	2363.64	2244.58	2338.11	2210.20
3	WSPT	2254.54	2260.11	2251.04	2234.70
	WCOVERT	2727.31	2592.26	2729.23	2573.43
	ATC	2564.14	2480.55	2558.27	2474.80
	WMOD	2563.71	2517.14	2609.77	2507.80
	CR+SPT	2409.50	2370.35	2421.75	2368.38
	S/RPT+SPT	2466.06	2410.22	2463.66	2386.53
4	WSPT	2128.72	2241.03	2335.81	2390.62
	WCOVERT	2843.91	2865.37	3262.29	3145.73
	ATC	2389.79	2338.61	2760.31	2688.10
	WMOD	2473.87	2562.83	2858.42	2860.01
	CR+SPT	2329.03	2460.30	2622.19	2652.83
	S/RPT+SPT	2367.50	2469.92	2693.79	2713.51

5.5 Conclusions

In the first part of this section we presented a new approximation algorithm, MULTIPCP, for the job-shop makespan problem. The key idea is, instead of solving the makespan problem directly, we repeatedly solve its dual problems with ready times and deadlines. By using Shifting Bottleneck as a comparison base, we have shown very competitive performance with MULTIPCP on two sets of problems previously published in the literature.

Since PCP is also fairly effective for the job-shop problems with nonzero ready times, general deadlines, and sequence-dependent setups, the approximation algorithm MULTIPCP can also be applied to the makespan problems with such kinds of structure with little difficulty. This can be considered as another advantage provided by the MULTIPCP algorithm.

We have also proposed and developed another new approximation algorithm for improving the mean weighted-tardiness performance for dispatching rules. The algorithm is base on the framework provided by the PCP procedure and is integrated with simple dispatching rules. This integration has proved to be able to provide global, useful information about the most promising precedence constraints whose interchanges will most likely generate better solutions.

The proposed procedure, referred to as WTPCP, was not only effective at reducing mean weighted tardiness but could also achieve significant reduction in number of tardy jobs under a variety of shop conditions.

Future research will focus on the investigation of more sophisticated *Tard* and *Slack* metrics and the development of new criticality indices. Possible extensions of WTPCP to problems with different objective functions or problems with different configurations like parallel machines or sequence-dependent setups will also be addressed.

References

[1] Adams, J., Balas, E., and Zawack, D., "The Shifting Bottleneck Procedure for Job Shop Scheduling," *Management Science*, 34, 3 (1988), 391 - 401.

[2] Akers, S. B. and Friedman, J., "A Non-Numerical Approach to Production Scheduling Problems," *Operations Research*, 3, 4 (1955), 429 - 442.

[3] Akers, S. B., "A Graphical Approach to Production Scheduling Problems," *Operations Research*, 4, 1956, 244 - 245.

[4] Anderson, E. J. and Nyirenda, J. C., "Two New Rules to Minimize Tardiness in A Job Shop," *Int. J. Prod. Res.*, 28, 12 (1990), 2277 - 2292.

[5] Applegate, D. and Cook, W., "A Computational Study of the Job-shop Scheduling Problem," ORSA J. Comput., 3, 2 91991), 149 - 156.

[6] Balas, E., Lenstra, J. K., and Vazacopoulos. A., "The One Machine Problem with Delayed Precedence Constraints and Its Use in Job Shop Scheduling", Graduate School of Industrial Administration, Carnegie Mellon University, #MSRR-589(R), 1993.

[7] Baker, K. R. and Kanet, J. J., "Job Shop Scheduling With Modified Due Dates," *Journal of Operations Management*, 4, 1983, 11 - 22.

[8] Baker, K. R., "Sequencing Rules and Due-date Assignments In A Job Shop," *Management Science*, 30, 9 (1984), 1093 - 1104.

[9] Carroll, D. C., "Heuristic Sequencing of Jobs with Single and Multiple Components," Ph.D. Thesis, Sloan School of Management, MIT, 1965.

[10] Erschler, J., Roubellat, F., and Vernhes, J. P., "Finding Some Essential Characteristics of the Feasible Solutions for a Scheduling Problem", *Operations Research*, 24, 1976, 772 - 782.

[11] Fisher, H. and Thompson, G. L., "Probabilistic Learning Combinations of Local Job-shop Scheduling Rules," J. F. Muth, G. L. Thompson (eds). *Industrial Scheduling*, Prentice-Hall, Englewood Cliffs, N. J., 1963.

[12] Florian, M., Trepant, P., and McMahon, G. B., "An Implicit Enumeration Algorithm for the Machine Sequencing Problem," *Management Science*, 17, 12 (1971), B-782 - B-792.

[13] Johnston, M. D. 1990. SPIKE: AI scheduling for NASA's Hubble Space Telescope. In *Proc. 6th IEEE Conference on AI Applications, Santa Barbara, CA.*

[14] Keng, N. and Yun, D. Y. Y. 1989. A planning/scheduling methodology for the constrained resource problem. In *Proc. IJCAI-89*, Detroit, MI.

[15] Lawrence, S., Supplement to "Resource Constraint Project Scheduling: An Experimental Investigation of Heuristic Scheduling Techniques," GSIA, Carnegie-Mellon University, 1984.

[16] Minton, S., Johnston, M. D., Philips, A. B., and Laird, P., "Solving Large-Scale Constraint Satisfaction and Scheduling Problems Using A Heuristic Repair Method," In *Proceedings of the Eighth National Conference on Artificial Intelligence*, Boston, MA. 1990.

[17] Muscettola, N. 1993. Scheduling by Iterative Partition of Bottleneck Conflicts. In *Proc. 9th IEEE Conference on AI Applications*, Orlando, FL.

[18] Pasch, k. A., "Heuristics for Job-Shop Scheduling", Ph.D. Thesis, Artificial Intelligence Laboratory, MIT, 1988.

[19] Sadeh, N., "Look-Ahead Techniques for Micro-Opportunistic Job Shop Scheduling," Technical Report CMU-CS-91-102, School of Computer Science, Carnegie Mellon University, 1991.

[20] Smith, F. S. and Cheng, C., "Slack-Based Heuristics for Constraint Satisfaction Scheduling," In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, Washington, D.C. July, 1993, 139 - 144.

[21] Taillard, E., "Benchmarks for Basic Scheduling Problems," *European Journal of Operational Research*, 64 (1993), 278 - 285.

[22] Vepsalainen, A. P. J. and Morton, T. E., "Priority Rules for Job Shops with Weighted Tardiness Costs," *Management Science*, 33, 8 (1987), 1035 - 1047.

[23] Zweben, M., Deale, M., and Gargan, R. 1990. Anytime Rescheduling. In *Proc. DARPA Workshop on Innovative Approaches to Planning, Scheduling and Control*, Morgan Kaufmann Pub.

6. Distributed Constraint Satisfaction through Constraint Partition and Coordinated Reaction

Many problems of theoretical and practical interest (e.g., parametric design, resource allocation, time-dependent scheduling) can be formulated as constraint satisfaction problems. Informally, a constraint satisfaction problem (CSP) is defined by a set of variables, each of which takes its value (is instantiated) from a given domain, and a set of constraints that restrict the admissible variable instantiations. To find a solution to a CSP means to find an assignment of values (an instantiation) for all variables, such that all constraints are satisfied. Recent work in DAI has considered the distributed constraint satisfaction problem (DCSP) [15, 6, 13] in which variables of a CSP are distributed among agents. Each agent has a subset of the variables and tries to instantiate their values. Constraints may exist between variables of different agents and the instantiations of the variables must satisfy these inter-agent constraints. Different models of assigning variables to agents have been investigated. In [15] each agent is responsible for instantiating a single variable, while in [13], each agent is responsible for a subset of variables, In these approaches, each agent was responsible for checking that all constraints involving the values of variables under its jurisdiction were satisfied, or identifying and resolving any constraint conflicts through asynchronous backtracking. Variables were instantiated in some order, according to a static ([15]) or dynamic ([13]) variable and value ordering, and the final solution was generated by merging partial instantiations that satisfied the problem constraints.

In this paper, we present an approach, called Constraint Partition and Coordinated Reaction (CP&CR), in which the set of agents is partitioned into agent subsets according to the types of constraints present in the DCSP. The fundamental characteristics of CP&CR are: (1) divide-and-conquer with effective coordination (2) avoid sophisticated inter-agent interactions and rely on collective simple local reactions. CP&CR divides a Constraint Satisfaction Problem into several subproblems, each of which concerns the satisfaction of constraints of a particular type. Enforcement of constraints on variables within a subproblem is assigned to a dedicated local problem solving agent which revises variable instantiations so that its own constraint type is satisfied. Since each variable may be restricted by more than one constraint, this means that the instantiation of a variable may be changed by different local problem solving agents. Each agent is iteratively activated and examines local views of a current solution. If it does not find any conflicts in the current iteration, it leaves the current solution unchanged and terminates its own activation. If it does find local constraint violations, it changes the instantiation of one or more variables. A final solution is an instantiation of all variables that all agents agree on, i.e. it does not violate any constraints.

The agents are unaware of each other's presence and constraints. Since constraint enforcement (change in the instantiation of certain variables) by a local problem solver may result in violations of constraints of other agents, the effectiveness of our approach requires coordination between local problem solvers on how they instantiate and revise the instantiation of variables to satisfy their own constraints. The exchange of coordination information helps the agents hedge against the myopia that is implied by the local nature of their problem solving. Experimental results reported in section 5 show the effectiveness and utility of various types of coordination information.

The domain of application of the methodology is job shop scheduling, one of the difficult constraint satisfaction problems. Job shop scheduling deals with allocating a limited set of resources to a number of activities (operations) associated with a set of orders (jobs). Job shop scheduling is a well-known NP-complete problem [7, 5]. Constraint-based approaches have been applied to the scheduling problem with very good results [4, 12, 11]. CP&CR views each activity as a *variable*. A variable's *value* corresponds to a reservation for an activity. A reservation consists of a start time and the set of resources needed by the activity. The dominant constraints in job shop scheduling are *temporal activity precedence* and *resource capacity* constraints. The temporal precedence constraints along with a job's release date, due date and activity durations restrict the set of acceptable start times for each activity. The capacity constraints restrict the number of activities that can use a resource at any particular point in time and create conflicts among activities that are competing for the use of the same resource at overlapping time intervals. The goal of a scheduling system is to produce schedules that respect the problem constraints, i.e. release and due dates, as well as temporal relations and resource capacity constraints.

In contrast to approaches [12, 11] that utilize incremental construction of partial schedules to produce a complete schedule, our approach first builds an initial schedule that possibly contains constraint violations and *incrementally revises* it to produce a conflict-free schedule. The revisions are made by specialized agents, each of which has a local view of a variable and can change the value (the start time) of the variables under its jurisdiction. Agents are of two types: Resource Agents that are responsible for enforcing resource capacity constraints and Order Agents, responsible for enforcing temporal precedence constraints. In this way, each variable is manipulated by a Resource Agent and an Order Agent. Schedule revision is the result of coordinated local reactions of the specialized constraint agents. The approach has been implemented in a system called CORA (COordinated Reactive Agents). Experimental results, presented in section 5 on a set of benchmark problems attest to the effectiveness of the approach as compared with other constraint-based scheduling methods.

6.1. Related Work

Approaches based on opportunistic heuristic search [4, 12, 11] generate schedules by opportunistically focusing attention to promising parts of the search space (e.g. bottleneck resources) and assigning one value to a variable at a time. Typically, they analyze the current situation, determine which is the next variable they should focus on, and then decide what is the best value to assign taking into consideration all involved constraints. After a variable has been instantiated, constraint propagation is performed to identify constraint violations that get resolved either by backtracking or by constraint relaxation. These approaches usually suffer intensive computational overhead. Constraint-posting approaches [2, 1, 10] generally do not commit to value assignments at the beginning but analyze the current situation, post additional constraints to exclude capacity conflicts, and then deduce variable assignments from the resulting network of constraints. CP&CR differs from the above approaches in that (a) it builds an initial, possibly flawed schedule and incrementally revises it, and (b) it does not perform global constraint analysis. Each constraint is locally enforced by coordinated reactions of the constraint specialists.

Iterative schedule repair [8, 16] is similar to CP&CR in that the solution is generated by iterative revision of an initial rough solution. However, in these approaches, conflicts are reduced by global evaluation of current conflicts and centralized decisions on which conflict to resolve next are made. The work of [9] is closer to CP&CR in that revision heuristics that have been acquired through case-based learning are used for incremental local patching of an initial schedule. Each revision locally enforces all constraints. After each local repair, constraint propagation identifies constraint conflicts caused by the repair. CP&CR differs from this approach in that it reduces conflicts by distributed local conflict resolution where each type of constraint is considered and resolved separately.

Similar to the decentralized character of CP&CR, [13] reported a distributed scheduling system where each agent has many variables under its jurisdiction and is responsible for resolving both capacity and precedence constraints associated with those variables. In [3] a Distributed Asynchronous Scheduler (DAS) is reported which consists of a number of reactive/opportunistic agents in a hierarchical organization. Unlike CP&CR, local problem solvers in these systems are all sophisticated agents and they perform intensive computation for their interactions.

6.2. Distributed Scheduling by Constraint Partition & Coordinated Reaction

Scheduling constraints are partitioned into two categories: temporal precedence and resource capacity constraints. Within each constraint type, subproblems are formulated. Each subproblem is assigned to a separate agent. In particular, enforcing capacity constraints on a given resource is a subproblem that is under the responsibility of a Resource Agent; enforcing temporal precedence constraints within an order is a subproblem that is assigned to an Order Agent. Therefore, for a given scheduling problem, the number of subproblems (and the number of agents) is equal to the sum of the number of orders plus the number of resources.

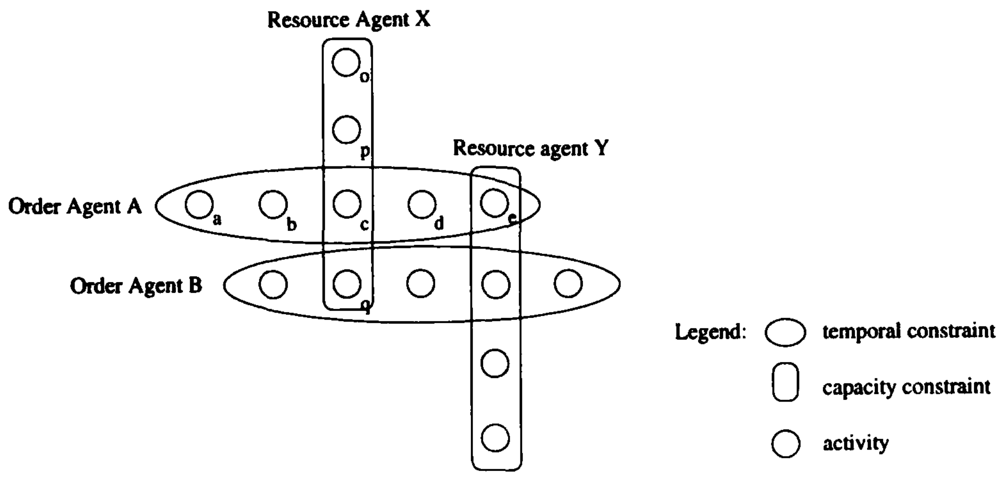


Figure 6-1: Constraint partition of a scheduling problem

Figure 3-1 shows a partial picture of the problem partition. Order Agent A is responsible for the satisfaction of temporal constraints on activities (a, b, c, d, e). Resource Agent X is dedicated to enforcing capacity constraints on activities (o, p, c, q). When these agents are activated, they can change instantiations of activities under their jurisdictions to satisfy their own constraints. Therefore, each activity (e.g. activity c) can be manipulated by both an Order Agent (e.g. Order Agent A) and a Resource Agent (e.g. Resource Agent X). Manipulation of activities by Order Agents may result in constraint violations for Resource Agents and vice-versa. Therefore, coordination between agents is crucial for prompt convergence on a final solution.

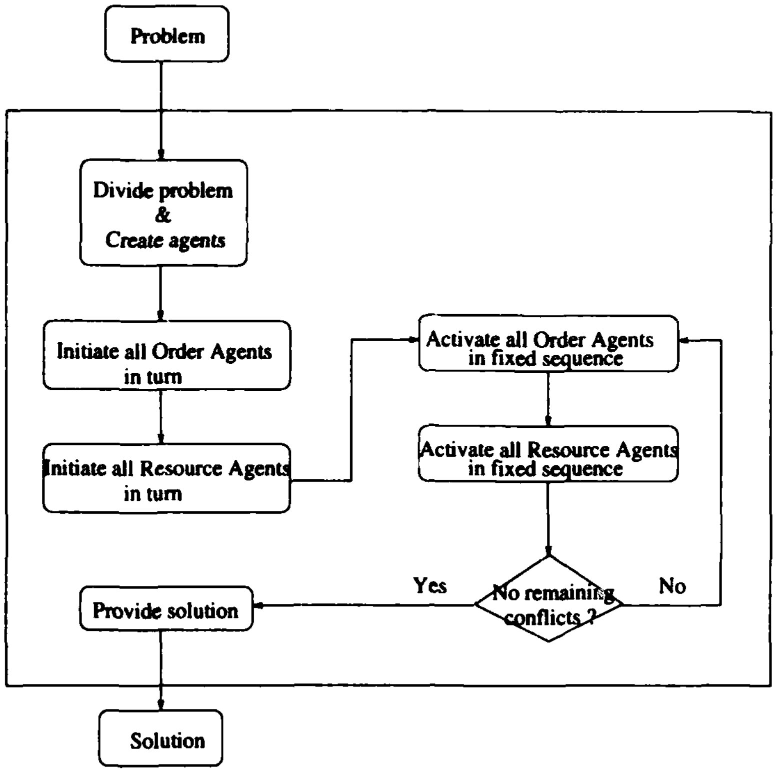

Problem
Divide problem & Create agents
Initiate all Order Agents in turn
Initiate all Resource Agents in turn
Activate all Order Agents in fixed sequence
Activate all Resource Agents in fixed sequence
No remaining conflicts ?
Yes
No
Provide solution
Solution

Figure 6-2: Control flow of CORA

In addition to the Order Agents and Resource Agents, the system includes a Manager agent. The Manager is a simple controller who performs the following tasks: (1) decomposition of the input scheduling problem according to resource and order constraints, (2) creation of the corresponding resource and order agents, (3) activation of the agents, and (4) outputting of the final solution when the system reaches quiescence, i.e. when no agent reacts to the current solution any more.

At system initialization, all order agents are activated first, followed by activation of all resource agents. At each subsequent iteration, the manager activates all order agents and all resource agents in turn. When an agent is activated, it revises the current values of the activities under its jurisdiction according to its local view. Processing stops when no agent has any constraint violations left. Figure 3-2 depicts the control flow of the system. Order agents are activated first because they calculate the *time boundary* for each activity under their jurisdiction (see figure 3-3). Time boundary information is associated with each activity and is used by the corresponding resource agent in instantiating or revising the activity's start time. The time boundary of an activity is defined as the interval between its earliest possible start time and its latest possible finish time. The boundary information for each activity is calculated only once during the initial activation of Order Agents and gets associated with the activity.

The initial solution, that is subsequently revised, is generated by the Resource Agents. The Resource Agents are activated after boundaries of all activities have been defined by the Order Agents. Each Resource Agent calculates the contention ratio for its resource by summing the durations of activities on the resource and dividing by the interval length between the earliest and latest time boundary among the activities. If this ratio is larger than a certain threshold, a Resource Agent announces itself as a Bottleneck Resource Agent. Activities under the

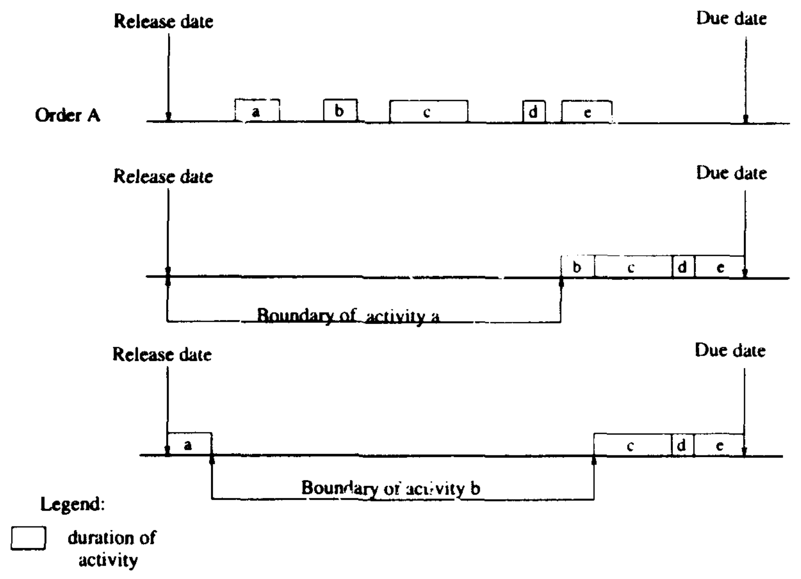


Figure 6-3: Calculation of temporal boundaries by Order Agent

jurisdiction of a Bottleneck Resource Agent are marked as Bottleneck activities.

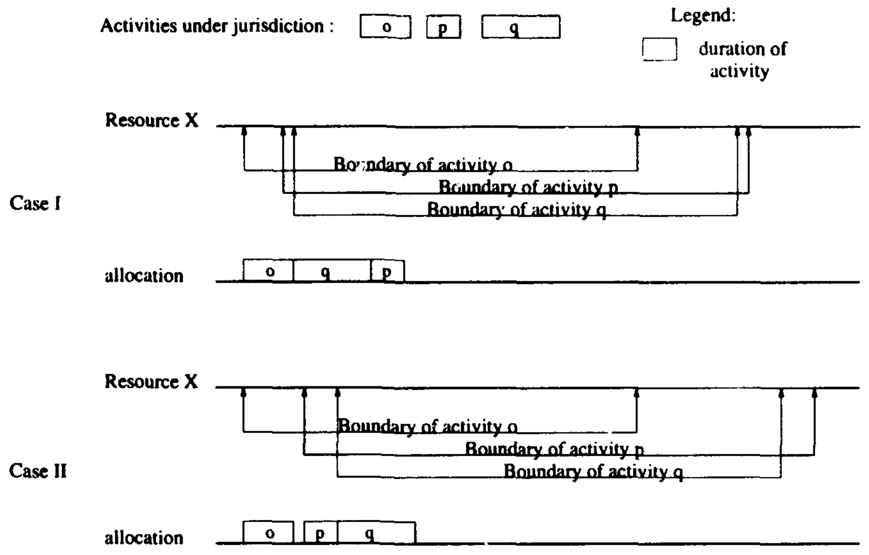


Figure 6-4: Initial allocation of resource intervals by Resource Agent

Each Resource Agent allocates resource intervals to activities under its jurisdiction according to their boundaries. Figure. 3-4 depicts different conditions of allocation of resource intervals based on sequence of activity boundaries. A Resource Agent allocates to each activity the earliest free interval on the resource. For example, activity o in figure 3-4 is allocated first because it has the earliest left boundary, and it gets the earliest interval. The next allocation depends on the next available start time (finish time of activity o) and the boundaries of remaining activities. If there are more than one activity eligible for allocation (left boundary is earlier than the next available start time), the next resource interval is allocated to the activity who has the earliest right boundary, such as activity q in Case I. If there is only one activity eligible, the resource interval is allocated to that activity at the next available start time. If there is no activity eligible, then the activity with the earliest left boundary is allocated to its earliest possible interval, such as activity p in Case II.

When there are multiple Resource Agents identifying themselves as Bottleneck Resource Agents, the Manager intervenes to choose the one with the highest resource contention ratio as the Primary Bottleneck Resource Agent and tells the others that they are Secondary Bottleneck Resource Agents. Secondary Bottleneck Resource Agents are then initiated again and re-allocate resource intervals according to a sequence of activities *with order-correspondence* to the sequence of activities used by the Primary Bottleneck Resource Agent (see Figure 3-5). This represents a coordination between Bottleneck Resource Agents and it plays an important role for solving scheduling problems with multiple bottleneck resources. Note that after this initial information exchange, the agents coordinate strictly according to local views.

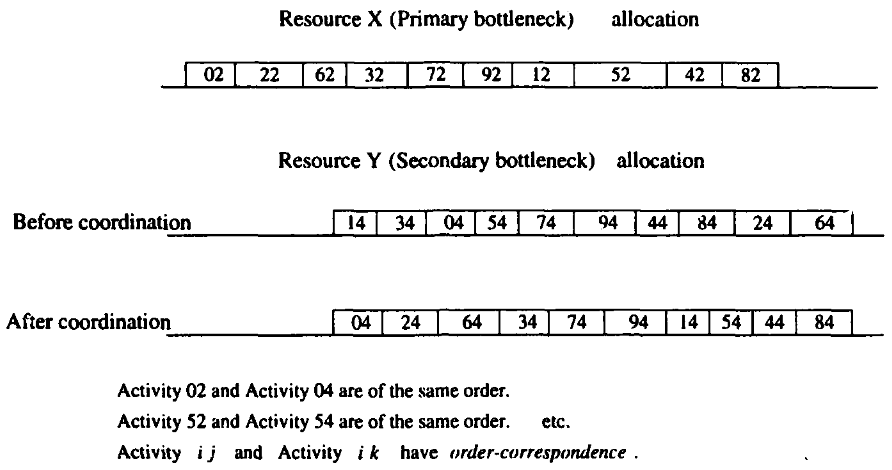

Resource X (Primary bottleneck) allocation
02 22 62 32 72 92 12 52 42 82
Resource Y (Secondary bottleneck) allocation
Before coordination
14 34 04 54 74 94 44 84 24 64
After coordination
04 24 64 34 74 94 14 54 44 84
Activity 02 and Activity 04 are of the same order.
Activity 52 and Activity 54 are of the same order. etc.
Activity i j and Activity i k have order-correspondence .

Figure 6-5: Coordination between Bottleneck Resource Agents

After the initial activation of Resource Agents, all activities are instantiated with a start time. This instantiation is the initial schedule that may subsequently be revised. The initial schedule does not contain resource capacity conflicts but it may contain temporal precedence constraint conflicts. Order Agents and Resource Agents coordinate through local reactions to the current solution so that their collective behavior results in a conflict-free final solution.

6.2.1. Agent Coordination

When activated, each agent reacts to the current instantiation of activities under its responsibility by going through an Examine-Resolve-Encode cycle (Figure 3-6). It first examines its local view of current solution, i.e. the values of the variables under its jurisdiction. If there are constraint violations, it changes activity instantiations to resolve conflicts (section 3.2). Since instantiation of an activity may be changed by an Order Agent and a Resource Agent back and forth, it is very important that they coordinate with each other in making changes to the current solution.

Since agents have no awareness of the existence of others, they do not communicate with each other directly. Instead, they coordinate by reading and writing coordination information on activities. Coordination information represents an agent's "view" on the partial current solution and is consulted when the agent needs to change the current instantiation to resolve its conflicts. After resolving conflicts, an agent *writes down its views* on current instantiations on each activity as coordination information. Coordination information *written* by an Order Agent on an activity

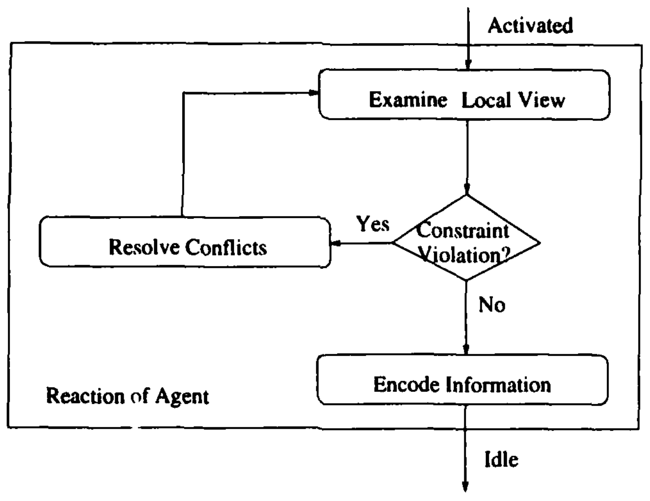


Figure 6-6: Agent reaction to current solution

is referenced by a Resource Agent, and vice-versa.

Coordination information provided by Order Agents (for Resource Agents)

Boundary: the interval between the earliest start time and latest finish time of an activity (as described in Figure 3-7). It represents the overall temporal flexibility of an activity. If a Resource Agent moves the activity outside this range, it will cause constraint violation for the Order Agent responsible for the activity. Activity boundaries are calculated only once during initial activation of Order Agents.

Temporal Slack: an interval between the current finish time of the previous activity and current start time of the next activity (see Figure. 3-7). It indicates the temporal range within which an activity may be scheduled without causing temporal constraint violations. (This is not guaranteed since temporal slacks of adjacent activities are overlapping with each other.)

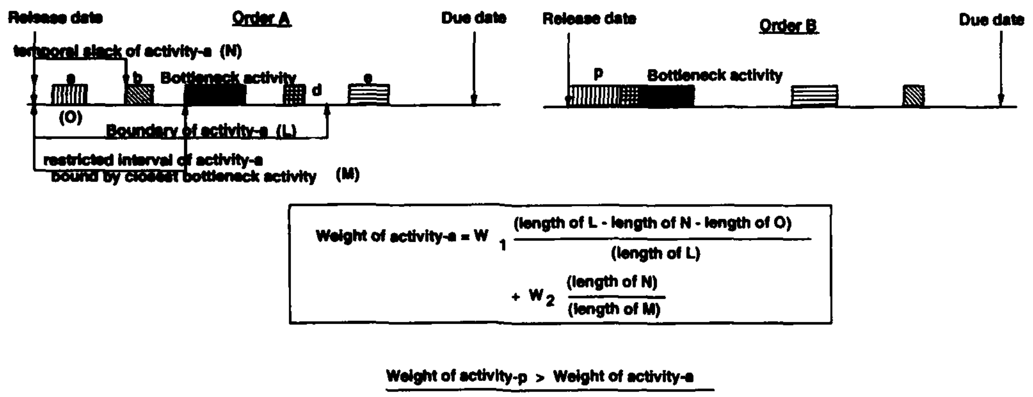

Release date
Order A
Due date
temporal slack of activity-a (N)
a
b
Bottleneck activity
d
e
(O)
Boundary of activity-a (L)
restricted interval of activity-a bound by closest bottleneck activity (M)
Release date
Order B
Due date
p
Bottleneck activity
Weight of activity-a = W1 (length of L - length of N - length of O) / (length of L) + W2 (length of N) / (length of M)
Weight of activity-p > Weight of activity-a

Figure 6-7: Temporal slack and weight determination

Weight: the weighted sum of relative temporal slack with respect to activity boundary and relative temporal slack with respect to the interval bound by the closet Bottleneck activities. It is a measure of the likelihood of the activity "bumping" into an adjacent activity, if it gets rescheduled. The higher the weight, the more likely it is that rescheduling the activity will cause conflicts. Therefore, a high weight represents a preference for not moving the activity (Figure

3-7).

Extra-weight: an additional measure of the importance of not moving the activity. There are three conditions for an activity to have extra-weight: (1) when an activity has bumped into a bottleneck activity and is moved to a new location, (2) when the number of times an activity has moved reaches a certain threshold, and (3) when a bottleneck activity is moved to a new location. At each of the three conditions, the extra-weight of this activity is set to a predetermined number.

Coordinated information provided by Resource Agents (for Order Agents)

Bottleneck Tag: a tag which marks that this activity uses a resource. This tag is put by a Bottleneck Resource Agent on all activities under its jurisdiction. It implies that the responsible Order Agent should treat this activity differently.

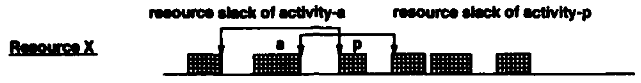


Figure 6-8: Resource slack

Resource Slack: an interval between the current finish time of the previous activity and the current start time of the next activity on the resource timeline (see Figure 3-8). It indicates the range of activity locations to which an activity can be moved without causing capacity constraint violations. (There is no guarantee on this since resource slacks of adjacent activities are overlapping with each other.)

Location Change: an index of whether the location on the timeline of this activity, set by an Order Agent, has been changed by a Resource Agent.

Change Frequency: a counter of how frequently the location of this activity set by an Order Agent is changed by a Resource Agent. It reveals a brief history of activity moves since agents do not keep track of previous locations of activities. High change frequency indicates that the locations where the Order Agent has moved the activity in the past caused capacity constraint violations which could not be resolved by continuing to move the activity. Therefore, the Order Agent should change locations of other activities (including bottleneck activity) to satisfy its constraints.

Figure 3-9 shows two groups of coordination information encoded on an activity. Coordination information encoded by an Order Agent is consulted by the corresponding Resource Agent, while an Order Agent consults coordination information encoded by the corresponding Resource Agent.

6.2.2. Reaction Heuristics

When an agent finds a constraint violation in an activity under its jurisdiction, it employs local reaction heuristics to resolve the violation. The reaction heuristics attempt to minimize the ripple effects of causing conflicts to other agents as a result of fixing the current constraint violation. Conflict minimization is achieved by minimizing the number and distance of activity moves.

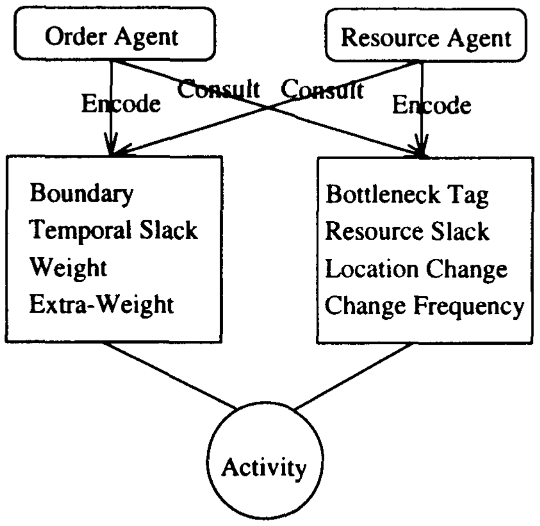
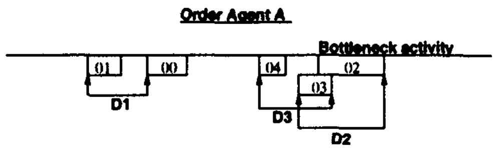


Figure 6-9: Coordination information

Order Agent

An Order Agent considers conflict resolution in the context of conflict pairs. A conflict pair involves two adjacent activities whose current locations violate the precedent constraint between them (see Figure 3-10). Associated with each activity is the distance it needs to be moved to resolve the conflict. If either one of the two activities is a bottleneck activity, the conflict pair is categorized as a bottleneck conflict pair, and is given a higher conflict resolution priority. An Order Agent treats a bottleneck activity as somewhat anchored. Since moving bottleneck activities causes severe conflict ripple effects, a bottleneck activity is moved only as a last resort.



		move
Bottleneck Conflict Pair :	activity 02	+ D2
	activity 03	- D2
Ordinary Conflict Pair :	activity 00	+ D1
	activity 01	- D1
Ordinary Conflict Pair :	activity 03	+ D3
	activity 04	- D3

Figure 6-10: Conflict identif'cation by Order Agent

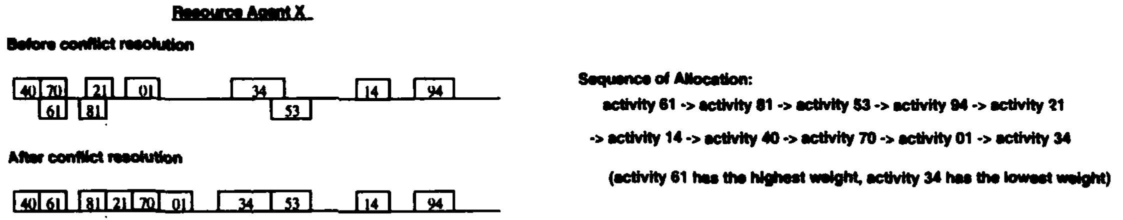
Conflict pairs are resolved one by one. To resolve a conflict pair, an Order Agent essentially determines which activity to move according to the distance associated with each activity. For a bottleneck conflict pair, if the change frequency of the non-bottleneck activity is below a threshold, it is the one to be moved. Otherwise, the bottleneck activity will be moved. To decide which activity to move in an ordinary (non-bottleneck) conflict pair, an Order Agent takes into consideration additional factors, such as moving feasibility of each activity, change frequency, resource slack and location change.

A Resource Agent is concerned with enforcing resource capacity constraints. If a capacity constraint is violated, the Resource Agent completely re-allocates the overallocated resource interval to the competing activities in such a way as to resolve the conflict and, at the same time, keep the location changes to each activity to a minimum.

Ordinary Resource Agent

An Ordinary Resource Agent re-allocates a resource interval to activities based on their weights.

Activities with higher weights get allocated first. When allocating a resource interval to an activity, an Ordinary Resource Agent tries the activity's current location first. If it has been preempted by another activity, an Ordinary Resource Agent looks for the two most adjacent intervals available (one left, one right) and chooses one for the activity according to its boundary and temporal slack. Since an activity's weight is a measure of the desire of the corresponding Order Agent to keep the activity at its current location, activity location decisions based on weight reflect group coordination. Figure 3-11 depicts how an Ordinary Resource Agent resolves conflicts. For example, the current location of activity 21 was preempted by activity 81 which has higher weight. Therefore, activity 21 gets an immediately adjacent interval.

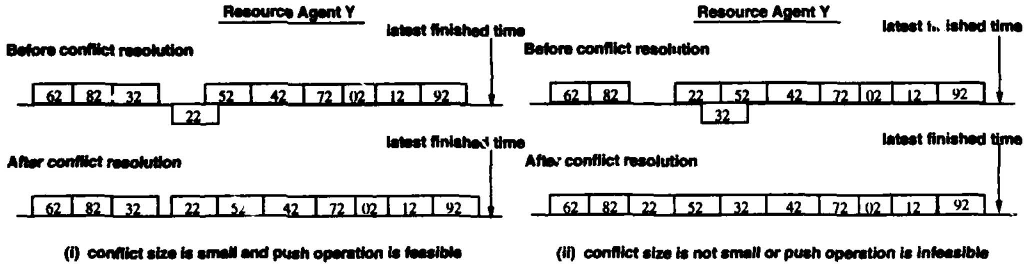


Resource Agent X
Before conflict resolution
40
70
61
21
81
01
34
53
14
94
After conflict resolution
40
61
81
21
70
01
34
53
14
94
Sequence of Allocation:
activity 61 -> activity 81 -> activity 53 -> activity 94 -> activity 21
-> activity 14 -> activity 40 -> activity 70 -> activity 01 -> activity 34
(activity 61 has the highest weight, activity 34 has the lowest weight)

Figure 6-11: Conflict resolution of Ordinary Resource Agent

Bottleneck Resource Agent

A Bottleneck Resource Agent has high resource contention ratio. This means that most of the time a Bottleneck Resource Agent does not allow resource slack between resource intervals (all resource intervals are immediately adjacent with each other). When activity moves occur, capacity constraint violations are very likely to occur. A Bottleneck Resource Agent considers the conflict size of capacity violations. The conflict size is the amount of overlap of activity reservations on the resource. If the conflict size is small and if right-shifting activities on the time line does not cause violations, then right shifting is performed (see Figure 3-12 (i)). Otherwise, it re-sequences activities according to their current locations, and then re-allocate resource intervals according to the new sequence of activities with no slack between each activity (see Figure 3-12 (ii)).



Resource Agent Y
latest finished time
Before conflict resolution
62
82
32
22
52
42
72
02
12
92
latest finished time
After conflict resolution
62
82
32
22
52
42
72
02
12
92
(i) conflict size is small and push operation is feasible
Resource Agent Y
latest finished time
Before conflict resolution
62
82
22
32
52
42
72
02
12
92
latest finished time
After conflict resolution
62
82
22
52
32
42
72
02
12
92
(ii) conflict size is not small or push operation is infeasible

Figure 6-12: conflict resolution of Bottleneck Resource Agent

6.3. Solution Evolution

Figure. 4-1 shows a solution evolution process for a relatively simple problem. In cycle 1, when Order Agents are first presented with the initial solution, there are totally 24 activities involved in temporal conflicts. After each Order Agent reacts to the current solution, Resource Agents find a total of 19 capacity constraint conflicts. Each Resource Agent reacts to the current solution. In the beginning of cycle 2, Order Agents only find 2 activities involved in conflicts. After the Order Agents' reactions, the Resource Agents again find 2 activities in conflicts. In

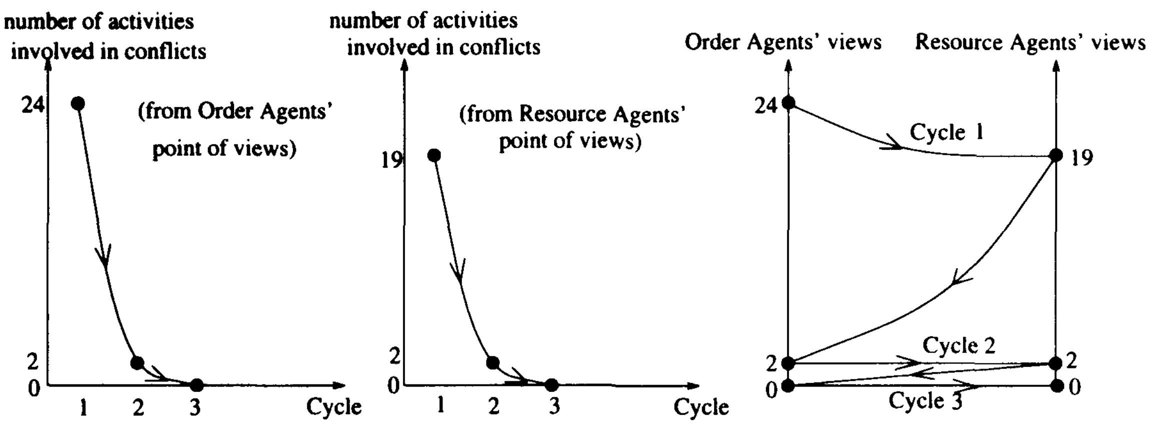


Figure 6-13: Solution evolution for a simple problem

cycle 3, both Order Agents and Resource Agents find no activity in conflict. A conflict-free solution has evolved.

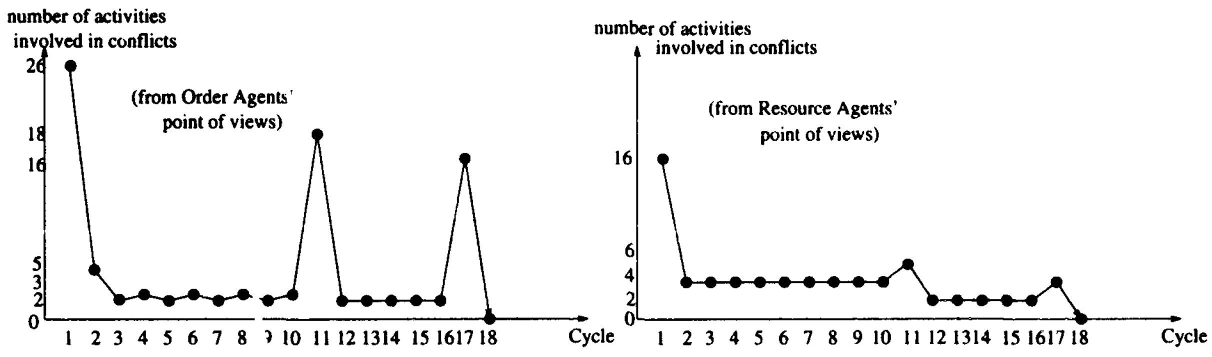


Figure 6-14: Solution evolution for a more difficult problem

Figure. 4-2 shows a solution evolution process for a more difficult problem. In the beginning of cycle 1, Order Agents find 26 activities involved in conflicts. After coordinated reactions of all agents, the number of activities involved in conflicts is reduced to 2 when Order Agents are activated in cycle 3. However, from cycle 3 to cycle 9 the solution evolution process is trapped in an oscillation. The number of activities involved in conflicts oscillates between 2 or 3 for Order Agents and 4 for Resource Agents. This indicates that activity values got switched back and forth by the respective agents because the agents could not find common ground for satisfying their respective constraints. However, since Order Agents and Resource Agents, have only local views, they are not aware of the situation. The crucial piece of coordination information that allows the agents to escape from the oscillation is the change frequency of activity location which serves as a history of activity moves. When the change frequency exceeds a heuristically determined threshold the Order Agent responsible for the variable instantiation moves a bottleneck activity. This initially increases the number of conflicts (as can be seen in the figure) but very soon it allows convergence to a conflict-free solution.

6.4. Experimental Results

We conducted two sets of experiments with CORA. At first, we compared CORA with three other state-of-the-art scheduling methods on a benchmark suite of schedu [illegible] blems. The results show that CORA outperformed the other methods in terms of number [illegible] lems solved and computational efficiency (CPU time). We also investigated the effects of coordination information in the system. We compared system performances on a set of coordination configurations ranging from no information to some information to adequate information. The results confirmed that coordination information facilitates successful and fast solution evolution.

The experiments were conducted on the suit of benchmark constraint satisfaction scheduling problems proposed in [11]. The benchmark consists of 6 groups of 10 problems, each of which has 10 jobs of 5 activities and 5 resources. Each job has a linear process routing which visits each one of the five resources. Activity sequences in the process routing are generated randomly, while bottleneck resources are visited after a fixed number of activities to further increase resource contention. Each group of problems differs in two respects: (1) spread of the release and due dates among jobs; (2) number of a-priori bottlenecks. The spread is controlled by varying the amplitute of the intervals within which release and due dates are generated. Three spread levels are introduced: wide (w), narrow (n), and null (0), i.e., both release and due date intervals are collapsed to single points. Aside from different spread levels of release and due dates, the benchmark also considered 1 and 2 a-priori bottlenecks conditions.

6.4.1. Comparison with other scheduling methods

Three other heuristic scheduling methods are compared with CORA on the benchmark: (1) Constraint Partition Scheduling [10], (2) Min-Conflict Iterative Repair [8], and (3) Micro-Opportunistic Search [11].

Constraint Partition Scheduling (CPS) constructs solutions by repeatedly identifying bottleneck conflicts and posting constraints to resolve them [10]. Analysis of resource capacity is based on a stochastic simulation. The final solution is deduced from the resulting constraint network.

Min-Conflict Iterative Repair (MIN-CONF) starts from an initial inconsistent solution and iteratively repairs it until a conflict-free solution is found. The initial solution is generated based on stochastic simulation. At each repair iteration, a variable is selected and a value is assigned to it based on the criterion of minimizing the number of remaining conflicts. If a solution is not found after a fixed number of iterations. a new initial solution is generated and the cycle repeats. Since MIN-CONF can run for a very long time if conflicts still exist, the maximum number of iterations allowed in the experiments was set to 5000.

Micro-Opportunistic Search (MICRO OPP) employs heuristic search with dynamic variable and value orderings. A solution is constructed by incremental extension of consistent partial value assignments. At each cycle, a variable is selected and assigned a value based on the heuristic orderings. Consistency is then enforced through the constraint network. The search backtracks when dead ends occur, e.g. the domain of the possible values of a variable becomes empty. MICRO OPP was run with two benchmark configurations - CHRON BKTRK, chronological backtracking and INTEL BKTRK, intelligent backtracking [14].

The performance results of CPS and MIN-CONF were reported in [10] in which these algorithms were each run 5 times for each problem because of their intrinsic random nature. CPU times for CPS and MIN-CONF were not available in [10]. However, we show CPU times based on unpublished recent work by Muscettola as their optimistic estimates. The performance results of MICRO OPP have been reported in [14].

	CORA	CPS	MIN-CONF	MICRO OPP	
				CHRON BKTRK	INTEL BKTRK
w/1	10	10 (0.96)	10 (0.36)	10	10
w/2	10	9 (0.89)	3 (0.33)	10	10
n/1	10	10 (0.94)	5 (0.44)	8	10
n/2	10	10 (0.92)	1 (0.40)	9	10
0/1	10	10 (0.82)	4 (0.25)	7	10
0/2	10	9 (0.91)	0	8	10
Total	60	58	23	52	60
AVG. CPU time	2.9 seconds	78.43 seconds	298.42 seconds	234.72 seconds	128.78 seconds

The above Table reports the number of problems solved and the average CPU time needed over all the benchmark problems for each technique. Each row represents different groups of the benchmark. For example, n/2 represents the group of problem with narrow spread and two a-priori bottlenecks. The numbers in the parentheses are the repeatability measures for CPS and MIN-CONF. Since CORA and MICRO OPP are deterministic, they don't have repeatability measures. Among all techniques, only CORA and MICRO OPP with INTEL BKTRK were able to solve all 60 problems.

The last row of the table shows the average CPU time over the entire benchmark for each technique. Note that the CPU times of CPS, MIN-CONF, and MICRO OPP are obtained from implementations in Common Lisp on a DEC 5000/200 workstation, while CORA is implemented in Allegro Common Lisp with CLOS on a SPARC IPX workstation. A DEC 5000/200 workstation runs feaster than a SPARC IPX workstation.

6.4.2. Comparison on Coordination Configurations

In order to investigate the effects of coordination information on the system's performance, we constructed a set of five coordination configurations.

- C0 represents a configuration in which the system ran with no coordination information at all. Without boundary information, when initially activated, the Resource Agents allocate resource intervals according to random sequences. When the Order Agents are activated, they resolve conflicts by randomly changing the instantiation of one of the two activities in each conflict pair. Similarly, the Resource Agents resolve conflicts based on random priority sequences.
- C1 represents a configuration in which only boundary information is available. The Resource Agents use this information for heuristic initial allocation of resource intervals. After the initial schedule is generated, no other information is available for conflict resolutions.
- C2 represents a configuration in which boundary and bottleneck tag information is available. The Resource Agents use the boundary information for heuristic initial allocation of resource intervals. The Order Agents use the bottleneck tag information to bias resolution of conflict pairs.

- C3 represents a near-complete configuration in which all coordination information is provided for the Resource Agents and Order Agents except coordination between Bottleneck Resource Agents on the initial allocation of resource intervals.
- C4 represents a complete configuration including initial coordination between Bottleneck Resource Agents.

C0 : No coordination information

C1 : Boundary (heuristic initial allocation)

C2 : Boundary + Bottlenck tag

C3 : Boundary + Temporal slack + Weight + Extra weight + Bottleneck tag + Resource slack + Location change + Change frequency

C4 : Boundary + Temporal slack + Weight + Extra weight + Bottleneck tag + Resource slack + Location change + Change frequency + Coordination between Bottleneck Resource Agents

Overall Performance	Coordination configuration				
	C0	C1	C2	C3	C4
No. of Porb. Solved(Avg.)	8.0	15.8	36.3	59	60
Avg. Cycle	33.3	34.8	24.7	6.6	5.8

Table 5-2
Comparative performance between coordination configurations

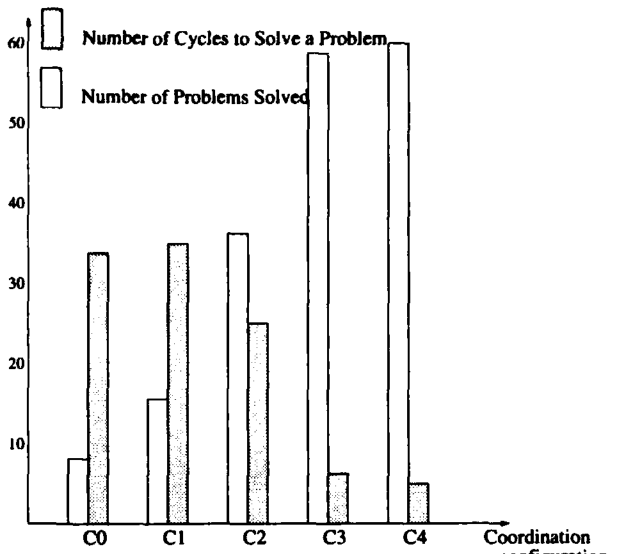


Figure 5-1.
Comparative performance graph between coordination configurations

The above Table and Figure show the comparative performance of different configurations on the suite of benchmark problems. In the table, the additional coordination information for each configuration is underlined. The number of cycles that the system was allowed was limited to 100. If there were still conflicts at cycle 100, the system gave up solving the problem. Since system operations in C0, C1, and C2 have random nature, they were ran on each problem 10 times. The numbers reported are the average number, e.g. 15.8 out of 60 problems were solved means that there were 158 successful runs among 600 (10 runs for each problem). C3 and C4 are deterministic and for these each problem was tried only once. The results show that more coordination information enables the system to solve more problems within fewer cycles. C0 was only able to solve 8 problems with an average of 33.3 cycles for solution evolution (4.7 CPU seconds), while C4 was able to solve all 60 problems with an average of 5.8 cycles (2.9 CPU seconds). A slight increase in the average cycles in C1 (compared to C0) stems from the fact that while C1 was able to solve twice the number of problems than C0 due to boundary information during initial allocation, it had no other advantages over C0 to resolve subsequent conflicts. With additional information, C2 was able to solve double the number of problems solved in C1 in fewer cycles. C3 solved almost all 60 problems in considerably fewer (6.6) cycles than C2. Only one problem, which is a 2-bottleneck problem, was unsolved by C3 within 100 cycles. By adding initial coordination between Bottleneck Resource Agents, C4 solved all 60 problems in 5.8 cycles. The results show the utility of coordination information.

As shown in Figures 4-1 and 4-2, the number of activities involved in conflicts in each cycle typically drops very fast within the first few cycles. After the drop, the problem solving process either solves the problem immediately or encounters a number of oscillations before finally solving the problem. Figure 5-1 shows, for different coordination configurations, the overall problem solving processes in terms of the number of activities involved in conflicts at each cycle. As the coordination information increases, the shape of the curve indicates a steeper drop in the number of conflicts in fewer cycles. Curves for deterministic C3 and C4 have peaks at cycle 5. This reveals that when the problem was not solved within the first few cycles, an escape from oscillation typically occurred.

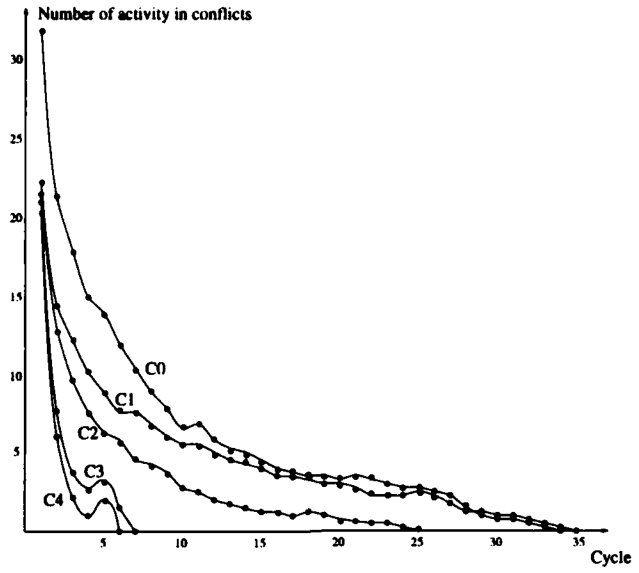


Figure 6-15: Comparison of successful solution evolution among different coordination configurations

6.5. Conclusions

We have presented an approach to distributed constraint satisfaction based on partitioning the problem constraints into constraint types. Responsibility for enforcing constraints of a particular type is given to specialist agents. The agents coordinate to iteratively change the instantiation of variables under their jurisdiction according to their specialized perspective. We demonstrated the effectiveness of the approach in the domain of job shop scheduling. Experimental results on a suite of benchmark problems showed that the approach outperformed other methods. The power of our approach stems from the types of coordination information that the agents utilize.

References

[1] J. Adams, E. Balas, and D. Zawack.
The Shifting Bottleneck Procedure for Job Shop Scheduling.
Management Science 34, 1988.

[2] D. Applegate and W. Cook.
A Computational Study of Job-Shop Scheduling.
Technical Report CMU-CS-90-145, School of Comoputer Science, Carnegie-Mellon University, 1990.

[3] Peter Burke and Patrick Prosser.
A Distributed Asynchronous System for Predictive and Reactive Scheduling.
Technical Report AISL-42, Department of Computer Science, University of Strathclyde, October, 1989.

[4] M.S. Fox and S.F. Smith.
ISIS: A Knowledge-Based System for Factory Scheduling.
Expert Systems 1(1):25-49, 1984.

[5] M.R. Garey and D.S. Johnson.
Computers and Intractability: A Guide to the Theory of NP-Completeness.
Freeman and Co., 1979.

[6] Huhns, M., Bridgeland, D.
Distributed Truth Maintenance.
In *Proceedings of the 10th International Workshop on DAI.* Bandera, Texas, 1990.

[7] E.L. Lawler, J.K. Lenstra, and A.H.G. Rinnooy Kan.
Recent Developments in Deterministic Sequencing and Scheduling: A Survey.
In M.A.H. Dempster, J.K. Lenstra, and A.H.G. Rinnooy Kan (editor), *Deterministc and Stochastic Scheduling.* Reidel, 1982.

[8] S. Minton, M.D. Johnston, A.B. Philips, and P. Laird.
Solving Large-Scale Constraint Satisfaction and Scheduling Problems Using a Heuristic Repair Method.
In *Proceedings of the Eigth National Conference on Artificial Intelligence.* 1990.

[9] Kazuo Miyashita and Katia Sycara.
Case-Based Incremental Schedule Revision.
In M. Fox and M. Zweben (editor), *Knowledge-Based Scheduling.* Morgan Kaufmann, 1993.

[10] Nicola Muscettloa.
Scheduling by Iterative Partition of Bottleneck Conflicts.
Technical Report CMU-RI-TR-92-05, Robotics Institute, Carnegie-Mellon University, 1992.

[11] Norman Sadeh.
Look-Ahead Techniques for Micro-Opportunistic Job Shop Scheduling.
Technical Report CMU-CS-91-102, School of Computer Science, Carnegie-Mellon University, 1991.

[12] Stephen F. Smith, Peng Si Ow, Claude Lepape, Bruce Mclaren, Nicola Muscettola.
Integrating Multiple Scheduling Perspectives to Generate Detailed Production Plans.
In *Proceedings 1986 SME Conference on AI in Manufacturing,* pages 123-137. 1986.

[13] Sycara, K., Roth, S., Sadeh, N., and Fox, M.
Distributed Constrained Heuristic Search.
IEEE Transactions on System, Man and Cybernetics 21(6):1446-1461, 1991.

[14] Yalin Xiong, Norman Sadeh, and Katia Sycara.
Intelligent Backtracking Techniques for Job Shop Scheduling.
In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning,* pages 14-23. 1992.

[15] M. Yokoo, T. Ishida, and K. Kuwabara.
Distributed Constraint Satisfaction for DAI Problems.
In *Proceedings of the 10th International Workshop on DAI*. 1990.

[16] M. Zweben, M. Deale and R. Gargan.
Anytime Rescheduling.
In *Proceedings of the DARPA Workshop on Innovative Approaches to Planning, Scheduling and Control*. 1990.

7. Reaching Informed Agreement in Multi-Specialist Cooperation

Advances in information technology and Artificial Intelligence have resulted in (1) the widespread use of knowledge-based systems in increasingly complex domains, (2) using several small systems in concert when their application domains overlap, and (3) integrating human and machine agents in cooperative problem solving. These trends argue for the development of knowledge-based systems in a distributed fashion where modules are constructed to interact intelligently and productively. The interconnected agents can solve problems cooperatively, work in parallel on common problems, increase the system's fault tolerance through redundancy, represent multiple viewpoints and share expertise. Thus, the focus of research in distributed knowledge-based systems and distributed AI has shifted from homogeneous to heterogeneous agents. Agents can vary along a variety of dimensions, such as capabilities, representation, and problem solving methods. Our current focus is the investigation of problem solving by cooperating specialists. Our approach is identifying variables and features that characterize cognitive processes and using them to create computational models. We believe that this methodology is useful in that (1) it can form the basis for designing machine aids to group decision making, (2) can guide the design of autonomous machine agents that are capable of effective coordination, and (3) by being based on investigations of human cognitive processes, the methodology could be a suitable tool for investigating the coordination and interface between human and machine agents in cooperative problem solving tasks.

Decision making in a team of specialists is fraught with difficulties [1, 11]. A major difficulty is that different specialists lack (1) a shared language for communication and (2) shared perceptions of the task. Evidence supporting this assumption comes from a variety of sources. Case studies of decision making in organizations (e.g., Bond's [3] Lockheed study of aircraft design) have found that specialists do not understand the details of each other's models and language, but through cooperation and interaction are somehow able to produce designs of very complex artifacts, such as aircraft. Bond [4] describes such organizational cooperation as occurring through a series of commitments. Our interest lies in investigating the process through which these commitments are formed.

Research in group decision making has found that the variability of approaches to decision making across groups appears to be even greater than across individuals [5]. The only consistently reported difference between successful and unsuccessful problem-solving groups has been that successful groups devote adequate time to *problem formulation* and *planning of meeting strategy*, whereas unsuccessful groups immediately begin to search for alternative solutions [9, 8]. In other words, successful groups spend adequate time to build shared vocabulary and mental models of the nature of the decision problem, strategies, significance of information, and participants' roles. Shared mental models have also been hypothesized by [2] and [7]. The importance of the formation of shared mental models has also been experimentally observed in time critical high-stress decisions such as the air crew emergencies studied by [15].

In DAI the problem of coordinating heterogeneous agents is receiving increased attention. A subset of approaches considers agents communicating through a common data structure, a blackboard, with restrictions in the overlap of the knowledge of each agent (e.g., [11]). Werkman's work [20] considers negotiation as the methodology for heterogeneous agents to arrive at agreement about a good design. In his approach, the agents reason monotonically to arrive at an acceptable compromise (through interactions or through a central arbitrator that resolves conflicts). Work on Distributed Truth Maintenance [13, 10] shares many of our concerns in belief revision of heterogeneous agents, but the emphasis is somewhat different. In Distributed Truth Maintenance models, the primary goal is maintaining forms of inter-agent consistency [10] rather than arriving at the highest payoff agreement through evaluation of alternatives. The focus of our model is to take advantage of the expertise available to the group in evaluating a decision without having to explicitly see or communicate that knowledge. This is similar to communicating meta-level control information in cooperative problem solving [6, 12]. In our model, the implicit hypothesis is that it is more efficient to share knowledge via evaluation than share the knowledge itself. Issues of consistency, and resolution of factuality are focused on the decision at hand rather than on the logical consistency of the agents' knowledge. Agreement on the ordering of the most favored alternative provides sufficient inter-agent consistency for the model to proceed.

Although evidence exists that the formation of shared mental models is a major factor in successful group decision making by humans, little has been done to characterize what "shared mental models" may be, the processes by which they are formed, or their role in group decision making. We argue that a computational model of the process of formation of mental models can become the basis for coordinating heterogeneous machine agents. In this paper, we focus on the characterization of the types of mental models that are active during cooperative decision making by heterogeneous agents, their role, and communications by which shared mental models of the task and the needed decisions are formed during problem solving.

The rest of the paper is organized as follows. Section 2 presents characteristics of shared mental models. The role of shared models in multispecialist cooperation is presented in section 3. Section 4 introduces the categories of mental models that form the building blocks of the architecture for fusing multiple expertise in evaluative decision making. Detailed presentation of the architecture is provided in section 5, and the interactions and model refinements are presented in section 6. Section 7 presents an illustrative example, and section 8 concluding remarks and the research evaluation methodology.

7.1. Characteristics of Shared Models

In the team of specialists situation, the agents are by definition heterogeneous, they do not share a common perception of the problem, or each others' expertise. Yet they have to make highly interdependent decisions with uncertain outcomes. A prerequisite to coordinating their actions is to coordinate their *thoughts* through the formation of shared mental models.

In particular, we present the following characteristics of shared models:

- Shared models act as a basis for inferring opportunities for cooperation
- Shared models form the basis for identifying another's needs resulting in helpful intervention
- Shared models act as a basis for inferring another agent's capabilities (resulting in the agent asking for help) and limitations, and taking them into consideration in communicating with other agents
- Shared models serve as a basis for knowing *what* to communicate (i.e. what will be

understood)

- Shared models act as basis for conflict resolution
- Shared models act as basis for optimal solution integration

The formation of shared mental models is an incremental process consisting primarily of information communication and successive updating and refinement of a sketchy understanding of the ramifications of the problem that the team is solving. The refinement process can be viewed as a kind of Kalman filtering, where both the predictive model and the observations are noisy. Various mental models are active during group decision making and interact to form shared mental models.

7.2. Problem Solving by Cooperating Specialists

The present model of decision making by a team of specialists extends earlier work [16, 17, 18, 19] characterizing group decision making as a negotiation and refinement process. A team of specialists is considered to be a group of individuals with common goals, each with highly specialized knowledge in a particular area but with less precise knowledge of other areas. The group's decision problem is to arrive at the best decision alternative that their joint expertise allows for a particular amount of communication. A normative group decision, therefore, is one which integrates relevant portions of this specialized knowledge within a common model. The key to this definition lies in the determination of relevance. We presume that in a group decision process of this sort, whether the agents are human or machine, it is neither feasible nor desirable to form a common model incorporating all of the group's expertise. Instead, a normative group decision is one generated by a process that efficiently elicits some small subset of the group's expertise which determines a decision as good or better than other possible interactions of comparable length. When sufficient behavioral data become available to characterize the departure of human agents from this normative standard we hope to incorporate it within a normative-descriptive model [14].

Group decision making by cooperating specialists can be viewed as a multi-agent task where each agent has (a) incomplete knowledge of the environment, (b) limited knowledge of the constraints and intentions of other agents, and (c) limited number and amount of resources that are required to produce a system solution. When these agents cooperate, they bring together multiple viewpoints and diverse knowledge on a single problem. Bringing together diverse knowledge is a source of robustness and balance. The team can solve problems that are beyond the scope of any of the individual members. Furthermore, the solutions are generated from a rich and varied body of knowledge which could provide a bigger set of good solutions to choose from and the potential for creativity. On the other hand, there are difficulties with resolving the conflicts that arise when trying to merge multiple goals, priorities, and evaluation criteria that are the results of individual expertise. This may result in misunderstandings, conflicts and solution suboptimalities. Because of the lack of appropriate expertise in all areas needed to solve the problem, and because of the presence of conflicting constraints, goals and possibly evaluation criteria, it is impossible for each agent to reach an optimal solution using only local information. Typically the decisions of one agent impact the decisions of another and vice versa. Thus, a computational model of cooperating specialists cannot simply model each agent. Rather it must augment an agent's problem solving process by including yet another model, the "shared model" that captures interactions and decision-coordination with the other agents.

Hypothesizing the existence of more than one agent model, gives rise to a variety of questions. How many agents models are active during problem solving? What is suitable information aggregation for inclusion in the "shared models"? How does a "shared model" interact with individual specialized knowledge? What is a suitable representation of individual expertise and "shared models"? What are the variables used for communication among group members?

7.3. Agent Models

Our hypothesis is that each agent:

- Has a model of his individual unique expertise, called the *"expert model"*, characterized by detailed knowledge about some particular aspect of the task.
- Has a naive understanding of aspects of the problem outside of his area of expertise, called the *"naive model"*. The naive model characterizes weak commonly held beliefs such as, "the more expensive a material is, the more durable it will be."
- Develops through interaction with other agents a more comprehensive model of the problem at hand, called the *"shared model"*, which incorporates elements of others' expertise. The "shared model" between two agents also defines a common vocabulary that the two agents can use to communicate in an intelligible way.

In general, shared models need not be identical across agents. For the group to make a common decision, the potentially different shared models that have been formed through interactions of subsets of agents must converge to a coherent[1] view of the global problem which captures the important variables and decisions that must be *coordinated*. Shared models evolve from the naive models by incremental modifications which make them conform to *justifications* and evaluations supplied by other agents. Agreement, however, is limited to those alternatives which have been considered. The naive model supplies both communication and inference capabilities by providing a common language, an inference mechanism for underdetermined evaluations, and an initial model for modification through communications. For modeling purposes we presume that a single naive model can be used to characterize the nonexpert knowledge of the agents. Pairwise commonalities will therefore decrease over the course of negotiation while agreement among evaluations will converge. The modified naive models which provide this convergence are what we refer to as *shared models.*

An agent's expertise consists of facts, constraints, their relations and utilities that lie in his specialty. The relations connecting the variables of his expertise can be arbitrarily complex. Figures 7-1 and 7-2 present a partial view of the expert models of the structural and aerodynamic perspectives in a turbine blade design task expressed in terms of a simplified model of relations and qualitative influences. For visual simplicity we have indicated in the figure only the edges connecting particular nodes, and the appropriate sign.

A path from node X to node Y in a qualitative influence graph constitutes a causal/justification chain that provides an explanation of the change in Y in terms of the change in X, assuming no

[1]Coherence requires consistency in private knowledge among agents. If our experts included both Keynesian and supply side economists, for example, a decision on tax policy in accord with the full expertise of the group would be incoherent and agent evaluations could not be expected to converge.

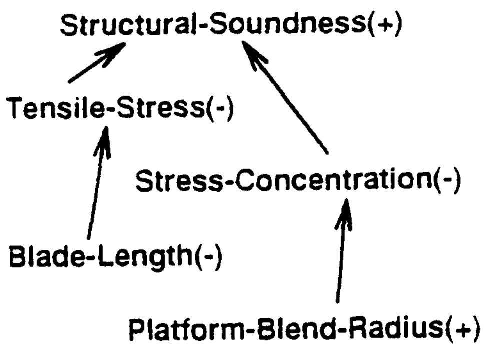


Figure 7-1: Partial expert model of structural engineering

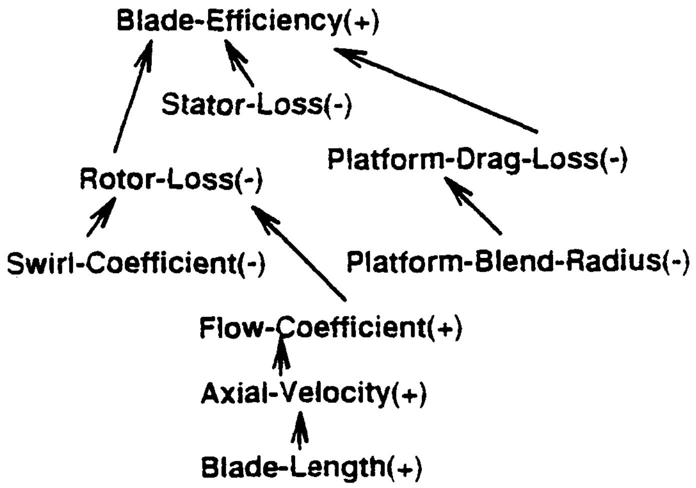


Figure 7-2: Partial expert model of aerodynamics

other change has occurred in the rest of the graph. For example, from the point of view of structural engineering, decreasing the length of the blade, Blade-Length(-), decreases tensile stresses, Tensile-Stress(-), which results in structural soundness, Structural-Soundness(+). In turn, an increase in structural soundness increases reliability, resulting in increased safety and contributing to increased marketability of the blade. By traversing its own graph an agent can find out which goals are supported by a set of design decisions, the utilities associated with particular design decisions and the justifications for the design decisions. This enables an agent to generate and/or select favorable design proposals, and generate justifications during problem solving.

At the beginning of problem solving, an agent's mental model consists of his expertise and a naive understanding of other specialties. In contrast to our earlier approach [16] where each agent has a model of the other agents, in this research the aim is for the agents to arrive at evaluative consensus without the burden of developing detailed models of other agents. Agents have minimal models of each other only in terms of knowledge of the kind of expertise that other agents have.

An agent's naive model consists of a set of decision variables and relations among them. These relations are at an aggregate level and are therefore only approximately accurate. For example, the naive model may say that production quality is proportional to production costs, unit sales inversely proportional to price, and willingness to pay (saleability) proportional to quality. As a result, the naive model predicts that saleability will be independent of design decisions balancing cost and quality. In other words the naive model is indifferent among design options. This evaluation made by the naive model is not accurate since it does not take into consideration precise relations between variables of the problem.

The naive model and an agent's expertise are connected through a set of relations that map variables within an agent's domain of expertise and those outside the domain of expertise. These mappings contain both accurate and approximate knowledge. The accurate knowledge is the set of relationships that map variables in the expert space to mediating aggregate (decision) variables that are within the area of agent expertise. For example, the structural engineering variables, such as tensile-stress and stress-concentration, constraints among them, their utilities and interrelationships are mapped into decision variables cost, price, and quality through mediating variables, such as structural soundness and reliability (see figure 7-1). These intermediate variables map to marketability, (see figure 7-5) which is an aggregate variable that is in the naive model of structural engineering but in the expert model of marketing.

The set of intermediate relationships that map the variables within an agent's specialty to the naive model (e.g., the set of relationships that map tensile-stress and stress-concentration to structural-soundness in figure 7-1) is available only to the particular agent and is not used for intra-group communication. On the other hand, although decision variables are used for communication, the relations between decision variables, within and outside the domain of expertise of an agent are incompletely known to him. For example the relation between structural-soundness (an aggregate variable within the domain of a structural engineer's expertise) and marketability is only approximately known to the structural engineer, since marketability is not one of the decision variables in his domain of expertise. This interaction of accurate and inaccurate knowledge may lead the agent to incorrect inferences.

Expert team decision making solves this problem through iteratively updating the naive models via the communication of the group members' expertise. During decision making, agent models appropriate for the task are composed by substituting parts of an expert model for relevant parts of the naive model. Therefore, an agent's evaluation of an alternative reflects both components. As group problem solving progresses, agent models are modified to incorporate expertise imparted by other agents. These modifications lead to the incremental building of shared models among sets of agents.

7.4. Representation of Agent Models

In this section we detail the representations that allow the expression of the expert, naive and shared agent models in a way amenable to computation. The overall model of an agent consists of a public *description space* used to characterize decision alternatives, a public *decision space* containing evaluative variables, and *naive* or *private* mappings linking descriptions to decisions. Examples of a decision alternative is a vector of description variables that have been instantiated to particular values in their domain. Decision variables are "aggregate" variables in that they refer to a decision alternative rather than a description variable (attribute). Agents are assumed to have a single naive common sense model of relations among decision variables, such as ''selling price, manufacturing cost, and quality covary'', which are represented in the decision space.

In more detail, the basic parts of the model are as follows:

1. *public and well defined description space* that consists of attributes of a decision alternative described in the public language l_0. In design, these attributes are attributes of the artifact, such as its dimensions, material, components, connections among components etc. Each description variable has a domain of values that respect appropriate constraints. For a turbine blade, a description consists of the vector of attributes/variables [root-radius, blade-length], and a design alternative could be [root-radius=35 in, blade-length=76 in].

2. *public and well defined decision space* modeled by influences among the decision variables represented as a directed acyclic graph. The language that describes the decision space is L_0. Edges of the graph linking two decision variables represent the relationship between them in terms of how one affects (positively or negatively) the achievement of the other. For example, in aircraft design, aerodynamic efficiency positively affects lower operation costs.

An influence $V_i \xleftarrow{+} V_j$ means that V_i increases with increasing V_j. $V_k \xleftarrow{-} V_l$ means that V_k increases with decreasing V_l. Examples of influences are reliability $\xleftarrow{+}$ structural-soundness, or saleability $\xleftarrow{-}$ price. In the current model we assume that relations among decision variables are directly proportional in their ranges so that a change in one will effect an increase or decrease of a corresponding size in another. As agent models are refined the (naive) influence of direct relations among decision variables decreases and is replaced by indirect relations through the joint influence of attributes in determining their values. For example, if quality and price were initially related only through the decision space they might later come to be partially related through the attribute "material" which has direct relations to both quality and cost. This phasing out of initial naive approximations in favor of more precise determination by attributes is what we mean by *refinement*. The model requires that values of decision variables be completely determined and that determination by attributes take precedence over determination by other decision variables. This allows the influences represented in the decision space to serve as an error term to the refinement process adjusting their weights as needed to preserve full determination. In a refined agent model in which no direct influences are left in the decision space, these effects could only be recovered as a relation among descriptions. Refinement involving closer approximation of $v_{i,j}$ and $B_{i,j}$ may continue even after every influence has been eliminated from the decision space.

In the current model, we assume that the relations among decision variables are linear with 45 degree slopes. This assumption simplifies propagation of new decision variable values and estimation of whether or not a new proposal increases profits. Figure 7-3 shows three such naive relations, production-cost $\xleftarrow{+}$quality, unit-sales $\xleftarrow{+}$quality and unit-sales $\xleftarrow{-}$unit-cost. For example, if a new proposal increases the product's quality by x%, then (assuming the relations in figure 7-3 and unit-cost $\xleftarrow{+}$production-cost) unit-cost must also increase by x% and unit-sales must also decrease by the same percentage. Since profit $\xleftarrow{+}$unit-sales, in this naive estimation, the new proposal leaves profits the same as the previous proposal. The initial naive relations are refined and updated as a result of the group's interactions. For example, the quality agent may say that while the change from stamping to machining increases quality by x%, unit-cost will decrease only by {x/2}%. This statement establishes a new relation between the attribute, manufacturing process, and the decision variables quality and production-cost replacing the previous naive relation between the two decision variables. The value of a decision variable is a function of one or more attributes in the description space and the influences of other decision variables. For example structural-soundness[root-radius=35, blade-length=76] = 8.5 (on an arbitrary scale 0 to 10).

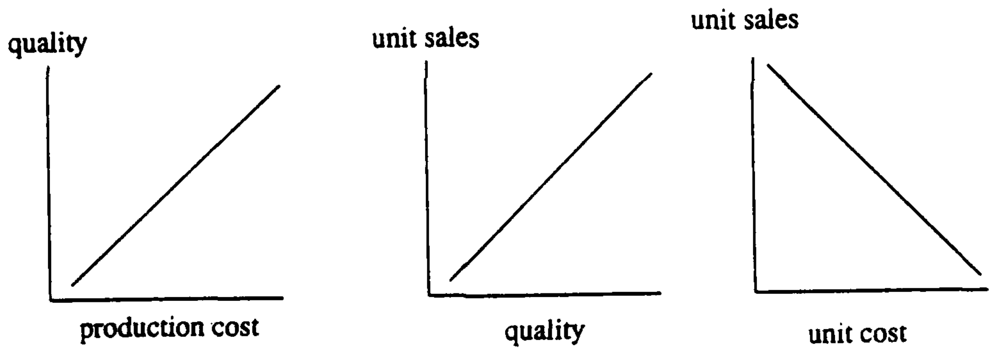


Figure 7-3: Naive relations

In the current version of the model, we make the assumption that the value of a decision variable represents the utility of the decision alternative with respect to the particular decision variable. The value that an agent assigns to a decision variable for a particular alternative may depend on its private knowledge. This representation provides both the multi-attribute utility values used to evaluate alternatives and a public representation relating decision variables used to determine the naive inferences needed to refine agent models.

3. *specialized/expert "black-box" knowledge* modeled as private functions of arbitrary complexity relating attributes to decision variables. The private knowledge of each agent α_i is expressed in its private language λ_i. In engineering design, specialized knowledge can be represented in terms of qualitative and quantitative relations and equations. This choice allows multiple attributes to jointly influence multiple decision variables creating "hidden paths" not represented in the public influence diagram.

4. ***naive mappings*** between description and decision variables modeled as publically defined functions $v_{i,k}$ (relating attribute a_k to decision variable V_i) and expressed in the language $\lambda_0 \in$ Λ_0 for non-expert agents. An attribute can be of relevance to more than one decision variable [illegible] the domain of a decision variable is a vector of more than one attributes. The relevance of a[illegible]tribute a_j to a decision variable V_i is its "contribution" and is expressed by a weight coe[illegible]ient $B_{i,j}$. Shared models are formed through the refinement of this naive knowledge. Refi[illegible]ents are plausible inferences defined as changes to naive portions of an agent's model. Refine[illegible]nts could change the coefficients $B_{i,k}$, the functions $v_{i,k}$ and through them the decision variable[illegible] which an attribute "contributes".

5. *expert mappings* between description and decision variables. The form of these mappings is determined by the agent's expert knowledge and is expressed in the agent's private language λ_i.

Figure 7-4 shows the architecture of the mental model of an agent. To calculate the value of a decision variable that is not within its area of expertise, an agent uses the publically known weighted sum of the functions $v_{i,k}$. Since refinements are limited to the naive portions of agents' models and result from public communications, these relations remain public under refinement. To calculate a decision variable within its domain of expertise, an agent uses its private expert knowledge. In the figure, the function ϕ expresses the expert mapping of the agent's private knowledge to decision variable V_3. Each agent's model is similar to that shown in figure 7-4, except that the "black-box" private knowledge would involve different attributes and decision variables.

Therefore, an agent's expert model consists of (a) the collection of qualitative and quantitative relations within its "black-box" along with (b) functions, such as ϕ that allow expert mappings between the "black-box" and decision variables. An agent's naive model consists of (a) the decision and description variables, (b) naive mappings within the decision space, and (c) naive mappings between description and decision variables.

In naive models, relations and contributions may be either indifferent (same contribution across attributes or same relation across attribute values) or ordered with respect to their contribution or relation to the decision variables. For example, in a naive model of the turbine blade root-radius and blade-length contribute equally to all decision variables, such as structural-soundness, cost etc. As another example, a naive model might hold unit sales which contributes to profit to be proportional to quality and inversely proportional to price with price proportional to quality in its decision space. If two materials, plastic and steel, were ordered with respect to quality, then this model would be indifferent to the choice because the contribution to profit of choosing steel (via quality and unit sales) is balanced by the adverse impact of quality (via price and unit sales) on profit. This sketchy knowledge of alternatives and their evaluation is shared (outside of individual areas of specialization) by all members of the group. The indifference of the naive model expresses the uncertainty of agents outside of their areas of expertise while still characterizing the common sense world knowledge they bring to the situation regardless of their specialization.

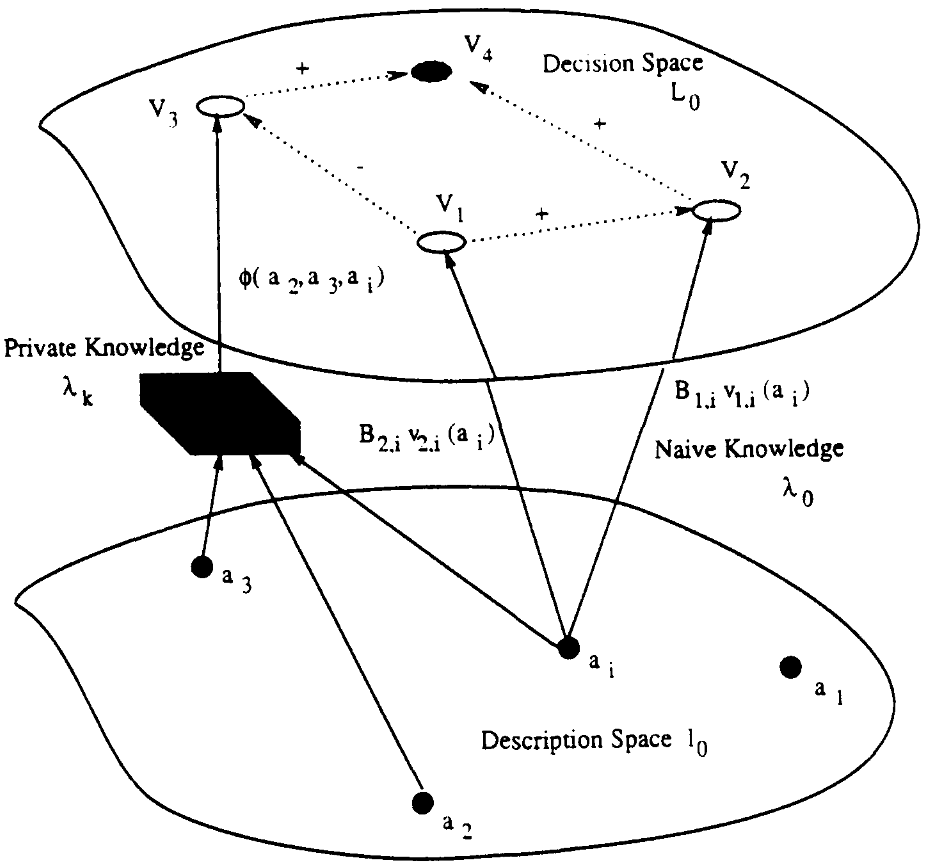


Figure 7-4: Agent Models

7.5. Communications

Each agent uses its private language λ_i to generate or evaluate proposals. However, communication among agents is restricted to the public languages L_0, l_0, and λ_0. This restriction allows agents to communicate their expertise only within the context of the group's problem and in terms that are intelligible to other agents in the group. So, although each agent's expertise is private to it, the common vocabulary is the medium for making public relevant portions or results of the expertise in the form of suggestions, justifications, and objections. In the turbine blade design example, terms such as Swirl-Coefficient and Axial-Velocity (see Figure 7-1) are private to the structural engineering agent. Terms such as Structural-Soundness and Blade-Efficiency belong to the common public vocabulary and are used for intelligible communication among the design agents. For example, the marketing agent understands the concepts of Blade-Efficiency and Structural-Soundness and how they relate to marketability, a decision variable within his area of expertise. In Figure 7-5, the shaded portions indicate the private expertise of the aerodynamics and structural agents, whereas the unshaded portion indicates terms to express goals and issues in the public vocabulary.

Communications are the means by which various parts of an agent's model get updated. A structural engineer justifying a proposal to thicken the turbine blade, for example, might report that increasing the root radius of a turbine blade by 5 inches would double its structural

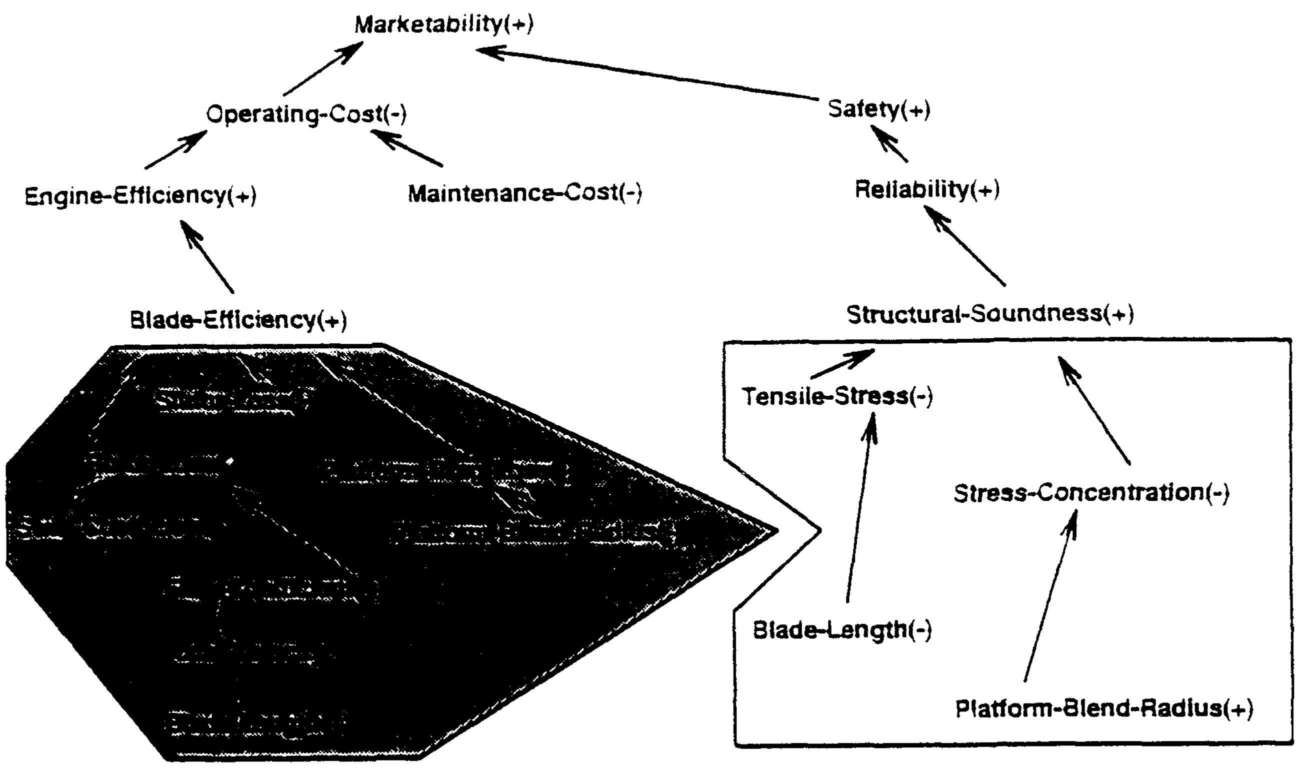


Figure 7-5: Partial view of private expertise and shared communication vocabulary

soundness. The engineering models behind this observation would remain private but the entire group could now benefit from his more valid estimate of the relation between a description variable, root radius, and a decision variable, structural soundness. Because other agents lack an understanding of the engineering judgement which underlies this pronouncement, they must continue to rely on naive inference (proportionality) to evaluate other alternatives. For example, another agent might infer that "if five inches doubles structural soundness then ten inches should quadruple it." Because a blade is a relatively uniform solid object, however, structural soundness is more closely proportional to cross sectional area (the square of its radius) making this an underestimate. This new misconception could only be rectified by further communications from the structural engineer. Because all agents' judgements are flawed in this way, they can only arrive at decisions reflecting their joint expertise through cycles of communication and updates to their individual naive models.

Information communicated by agents is classified as *evaluation* or *justification*. An expression of preference among alternatives is an evaluation. An expression relating attribute values to decision variables is a justification. Communications will often contain information of both sorts. The statement: "We should not change from alloy to composite materials because the manufacturing cost would be too high," for example, consists of the evaluation: profit(design-composite) < profit(design-alloy)[2] and the justification: manufacturing-cost(design-composite) > manufacturing-cost(design-alloy). Note that the evaluation is re-expressed in terms of the implicit and publically known relation between manufacturing cost and profit. This is done to clarify the status of evaluation as an ordering of alternatives with respect to the decision criterion rather than a relation between attributes and a particular decision variable. As suggested by the example, the distinction between these forms of information lies in the relation referenced rather than the apparent phrasing. The statement: "We should choose the highest quality design because profits depend on quality" contains no justification because it does not relate attributes to decision variables. This communication conveys, instead, the agent's "evaluation" of the contribution of quality to profits. In our model, the assumption of a common naive model makes this statement vacuous (devoid of information) since the "evaluation" is already known to other agents.

Factual statements such as: "Selection of steel instead of plastic would increase material costs by 30%" are justifications because they relate attribute and decision variable values even though they lack an evaluation to be "justified". To make an evaluation between the alternatives design-steel and design-plastic, as a result of hearing this statement, an agent must perform the inference material-cost(steel) > material-cost(plastic), material-cost and manufacturing-cost are related in a positive manner (+), manufacturing-cost and profits are related in an inverse (-) manner, hence profit(design-steel) < profit(design-plastic).

Justifications may be expressed in any of four forms, each of which conveys a different type of information about the relation between attributes, private knowledge, and decision variables. These forms are:

1. *value ordering-* The communication expresses an ordering on values of a decision variable as a function of attribute values. Value orderings have the form: relation(attribute_value-1,attribute_value-2) -> Order(v(attribute_value-1),v(attribute_value-2)), where $v_{i,j}$ is the function relating values of attribute a_i to decision variable V_j. The earlier statement: "We should not change from alloy to composite materials because the manufacturing cost would be too high," is an instance of value ordering because it orders the nominal values "alloy" and "composite" of the attribute, material, by their relation to the decision variable, manufacturing cost. Value ordering communications refine agent models by modifying function $v_{i,j}$.

2. *contribution determining-* The communication expresses the contribution of an attribute to a decision variable. Contribution statements may be either absolute or relative. The most common absolute contribution statements assert that for values under consideration an attribute makes no contribution to a particular decision variable. A statement of this sort would be: "The choice between composite and alloy materials will not affect reliability." An example of a relative contribution justification would be: "Choice of wheel attachments is more important to

[2]Note that since profit is the decision criterion, the statement profit(design-composite) > profit(design-alloy) is equivalent to utility(design-composite)> utility(design-alloy).

the reliability of a tricycle than choice of frame material." Contribution justifications do not provide information about the relation between values of attributes and decision variables but instead characterize the extent of the attribute's contribution. Contribution determining justifications refine agent models by modifying the weight $B_{i,j}$ which determines the contribution of attribute variable, a_i, to decision variable, V_j.

3. *value determining*- The communication expresses both the contribution and the relation between attribute values and values of a decision variable. Value determining expressions have the form: relation(attribute_value-1,attribute_value-2) -> V(attribute_value-1)=KV(attribute_value-2) where K is the ratio between the two aggregate values. The earlier statement: "Selection of steel instead of plastic would increase material costs by 30%" is an example of a value determining communication: material-cost(steel)=1.3material-cost(plastic). Value determining communications may affect either the function, $v_{i,j}$, its weight, $B_{i,j}$, or both. The refinement which occurs depends on the prior values of $B_{i,j}$, the range of $v_{i,j}$, and the range and weights of other attributes with non-zero contributions to V_i. These adjustments are made according to the calculus of propagating values of decision variables based on the linearity relations and their refinements, subject to the constraints:

$\max(V_j)= \sum_k B_{k,j} max(v_{j,k}(a_k))$

$C= \sum_k B_{k,j}$ where C is a constant reflecting the relative scaling of weights

4. *conjunct labeling* - Justifications may involve terms from an agent's private language. In these situations the public version of the term from the private language serves to "label" a relation involving multiple attributes and decision variables expressed in the public language. A communication of this sort about a proposed tricycle design might be: "We should change to a ball-bearing and race for the headset (attribute-1) if the frame is plastic (attribute-2) and the bearing is metal (attribute-3) because otherwise torsion will cause the bearing to slip and weaken the frame (private knowledge) making the tricycle unreliable (decision variable)". The justification in this case is the public expression: reliability(headset=ball-bearing-and-race & frame=plastic & bearing=metal) < reliability(headset=bearing & frame=plastic & bearing=metal) which is labeled by the private term "torsion". Labeled justifications are consistent with our contention that it is neither feasible nor desirable for specialists to develop detailed models of one another's expertise. In this example the technical meaning of the word "torsion" and its use to describe forces affecting mechanical devices remains private to the communicating agent. In the public communication the term "torsion" serves only to label a public expression relating these three attribute values to the decision variable, reliability. Conjunct labeling reifies private knowledge by expressing relations between multiple attributes/decision variables in public form for the values of some particular alternative. Agent models are refined by adding this new composite attribute to their description space.

Justifications are further classified as expert or naive. An expert justification is a justification which involves decision variables within an agent's domain of expertise. A rejection of a proposed turbine blade design by the structural engineer which referenced a change in the decision variable, structural soundness, for example, would be classified as an expert justification. If the structural engineer's justification for rejection had involved the decision variable, manufacturing cost, instead, it would have been classified as a naive justification.

Refinements are naive in much the same way as the influences in the decision space they replace. The model again assumes proportionality in the absence of more precise specification and uses the determination of weights and values in previously considered alternatives to anchor the evolving model. Functions $v_{i,j}$ and labeled conjuncts are learned in a piece-wise linear fashion and are revised to maintain fit to previous alternatives when contribution weights are adjusted. Orderings of alternatives enter the model as constraints which are converted to values in accordance with the proportionality and indifference assumptions of the model. These values, like the naive ones they replace, are treated as "second class citizens" by the refinement which anchors the evolving model to directly determined values. Because values of decision variables are computable as weighted sums of functions of attributes, evaluation of alternatives is always possible.

The decision making process can be seen as hill-climbing where the group starts with an initially proposed (perhaps randomly generated) decision alternative and iteratively adapts it to arrive at an improving best decision. This search is satisficing rather than optimizing. Because the distributed evaluation function is not fixed, both the evaluation of alternatives and the choice of alternatives to evaluate are determined by the history of interactions. Whether or not justifications are exhaustively exchanged (full expertise), evaluation depends jointly on alternatives already considered, degree of refinement of shared models, and the alternative itself. Within the model it is possible for a shallowly explored alternative to resurface later for evaluation at greater depth. This sort of behavior is not backtracking in a strict sense because the alternative has changed with respect to the agents' evaluations. As a conventional search problem, the model would require the power set of possible histories and alternatives as its search space. There is no guarantee that a group would not stop at a local maxima or even stop short of it if evaluation were sufficiently favorable. The problem addressed by this model is not how to escape local maxima but how to detect the hills which may be invisible to individual agents.

In our model, we assume that maximizing profits is the *decision criterion.* Each current decision alternative is taken as a baseline that must be improved by the next acceptable proposal (i.e. inferior proposals will be rejected). Each new proposed decision alternative contains at least one change in an attribute value, and possibly results in new values for a subset of the decision variables, or possible update of various mappings between variables in the description and decision space.

Under the restriction that only expert justifications lead to refinement, communications of sufficient length will cause all agent models to converge to a single evaluation for any particular alternative. Convergence follows from the anchoring of refinements to attribute and decision variable values associated with an alternative. Model refinement with respect to this alternative is simply an inefficient mechanism for exchanging these values. Even absolute convergence for a single alternative contributes little to improving decision making which requires agreement in the ordering of alternatives rather than precise agreement on the utility of any particular alternative. Our focus is not on the role of communication per se in providing accurate evaluations of particular alternatives but rather on the role model revision plays in directing search. As agent models are refined, their evaluative ordering on alternatives will change causing alternatives previously dismissed as infeasible to become practical while causing undesirable ones to become newly attractive, thus allowing proposing new alternatives which increase the value of the decision criterion. These revisions and reevaluations occur within individual agent models taking advantage of individual expertise and an improving approximation of problem relevant aspects of other agents' models. An exact control procedure for using agent models to direct search is not dictated by our model. Plausible choices include bidding among agents for proposing the next alternative (if private reevaluation results in many agents' estimation of differential increases in the value of the decision criterion), round robin selection of the next proposer, or joint partial specification of the next alternative by multiple agents, each of which specifies values for a subset of description variables.

7.6. An Example

Consider the decision situation for a team of specialists in a manufacturing enterprise tasked with concurrently engineering the design of a tricycle. The team objective is to arrive at a tricycle design that will maximize profits, under certain assumptions relating design attributes to cost, price, and ease of selling the tricycle. The group's goal may be expressed as:

profit = unit_sales x (unit_price - unit_cost)

Tricycle sales, price and cost can be expressed as functions of high level attributes, such as tricycle, performance, style, durability, ease of use, reliability, structural soundness etc. Since there is no precise mathematical model of design evaluation (there are too many variables that interact in non-linear and unpredictable ways) or design saleability, the group's goal of "optimizing the design" gets operationalized to "using profits as a decision criterion, find a design acceptable to all concerned agents". This is as optimal a decision as the group can give since it is a decision that takes into consideration the fused expertise of the group. The decision problem then is to evaluate alternative designs, negotiate on suggestions for design modifications and arrive at a design agreeable to all.

Let us suppose that there are three agents involved: a designer, a manufacturing agent and a sales agent. In his expert model, the designer knows the precise (true) relations among design attributes and decision variables such as performance, durability etc. For example, a designer's expert model predicts that high grade plastic makes the tricycle lighter thus leading to higher performance, whereas heavy steel tubing leads to lower performance. In terms of strength of the tricycle frame, braced and welded frame leads to higher frame strength, bolted the next highest, and integral the lowest. In terms of reliability, using cotter pins and caps to hold the rear wheels and pedals of a tricycle together results in much higher reliability (the designer's model may include precise equations or empirical results from which precise numbers could be derived), than using press-on caps, since press on caps are likely to start falling off after a short time of tricycle use.

Let us concentrate on the design choice of cotter pins versus press on caps. Before the group meeting, each agent has different evaluations of the two designs. Suppose that the designer's expert knowledge rates the cotter pin design twice as reliable as the press-on caps design. On the other hand, the precise relations between cost and other decision variables are not in the expert model of the designer. The designer's naive model considers that sales and cost are linear with reliability. This leads him to infer: (1) the cost of a cotter pin design will be double the cost of a press-on cap design, (2) the sales for a cotter pin design will be double the sales for a press-on design, and (3) he should be indifferent to the design choice. Similarly, the manufacturing engineer knows (from his expert knowledge) that drilling round stock to make the hole for the cotter pins is 3% more expensive than fitting the press-on caps. The expert model of the manufacturing agent does not contain precise knowledge of the relation between press on caps (or cotter pins) and reliability (or other high level design attributes). His naive model considers cost and sales linear with reliability. This leads him to infer: (1) a cotter design is 3% more expensive than a press-on design, (2) a cotter design will sell 3% more than a press on design and (3) he should be indifferent to the choice. The expert model of the sales agent predicts that *appearance of ruggedness* (for a device used by children, and which will suffer a lot of wear and tear) multiplies sales by a factor of three. The sales agent has no expert knowledge about either the relative cost for manufacturing cotter pins versus press-on caps or about relative reliability of the two designs. Thus, he is initially indifferent to the choice.

The group meets. Drawings for the two designs are displayed and discussed. The group interaction over the design choice of press-on caps versus cotters may proceed as follows:

Manufacturing Agent: Do we want to use press-on caps or cotters for the wheels and pedals? The drill press operations will add another 3% to manufacturing costs. (This is a value determining communication). [The design agent learns that cotter pin cost is only 3% more rather than double. He, therefore, updates the (naive) relations between fasteners (attribute relevant to decision variables), cost and reliability. On the other hand, the naive relation of linearity between reliability and sales predicts that the cotter design should almost double profits, thus furthering the group's goal of profit maximization and making the design agent strongly prefer the cotter design. The sales agent learns the relative cost of cotters, cost(cotters)=1.03cost(press-ons) and updates the cost decision variable].

Design Agent: In that case, I think we should use cotters. The press-on caps are likely to start falling off after 6 months to a year while the cotter pins will hold the wheels on for 10 years. [The manufacturing agent now learns that the cotter design is much more reliable, reliability(cotters)>10reliability(press-ons), and updates his model so that cost is now indirectly related to reliability through the fastener attribute. The sales agent learns that cotter design is much more reliable than press-on design and updates the relevant part of his naive model. However, his expert model tells him that it is the appearance of ruggedness that sells the product. By looking at the designs, the sales agent finds out that a buyer cannot see whether a cotter or a press-on has been used].

Sales Agent: I don't think we should use cotters. A buyer can't see that there is a cotter under the cap and therefore it has no effect on appearance. On the other hand, cotters are 3% more expensive.

The design and manufacturing agents substitute this direct relation between the fastener attribute and sales for the previous relation between decision variables and now agree because with this modification, their shared model predicts that the increased production cost associated with cotters will be detrimental to the goal of increasing profits because there is no offsetting influence of this form of reliability on unit sales.

7.7. Concluding Remarks

The importance of shared mental models has been recognized in the cognitive decision making literature [2, 7] but no investigation as to the process of forming shared mental models has been proposed to date. We have proposed the existence of a variety of mental models (the expert model, the naive model, shared models and a potentially common model) during group decision making and presented their interactions during the problem solving process. In addition, we have presented an initial approach to describing those models and the process by which they are formed. The shared mental models discussed in this paper define the nature and level of aggregation of the information necessary to be common knowledge to agents so that suitably coordinated decisions can be made.

The model presented in this paper is intended as a research tool for the study of group decision making by people and machines. To evaluate the research, we have planned a three stage approach. First, the multi-agent model will be implemented as a DAI system consisting of machine agents. We have planned a number of experiments of the system's performance under a variety of conditions (e.g. various restrictions on what gets communicated, differential sophistication of the models of others that each agent possesses, differential forms of the naive model). Second, controlled experiments with human subjects will be performed to compare the predictions of the model (as embodied in the behavior of the DAI system) with decisions made by the subject group in concurrent design tasks of simple artifacts, such as a children's tricycle. Each subject will be given a "cover story" including his/her naive and expert model, and experiments will be controlled by restricting the form and contents of communications. Third, based on the subject group's observed decision biases, a decision aid will be developed that compensates for the biases by referencing them to a progressively refined version of the naive model. The goal is to develop a domain independent decision aid that has knowledge *about the refinement process* rather than domain expertise. The aid uses knowledge about the refinement process to evaluate group communications that fall outside its model, provide suggestions, and make queries to point out the group's misconceptions.

References

[1] Adler, M.R., and Simoudis, E.
Integrating Distributed Expertise.
In *Proceedings of the 10th International Workshop on DAI.* Banderra, Texas, 1990.

[2] Athans, M.
The Expert Team of Experts Approach to Command-and-Control Organizations.
IEEE Control Systems Magazine September:30-38, 1982.

[3] Bond, A., and Ricci, R.
Cooperation in Aircraft Design.
In *Proceedings of the MIT-JSME Workshop on Cooperative Product Development.* Cambridge, Mass., 1989.

[4] Bond, A.
PROJECTS: A normative model of collaboration in organizations.
In *Proceedings of the 10th International Workshop on DAI.* Banderra, Texas, 1990.

[5] DeSanctis, G., and Gallupe R. B..
A Foundation for the Study of Group Decision Support Systems.
Management Science 33:589-609, 1987.

[6] Durfee, E.H.
A Unified Approach to Dynamic Coordination: Plannign Actions and Interactions in a Distributed Problem Solving Network.
PhD thesis, COINS, University of Massachusetts, 1987.

[7] Fischhoff, B.
Decision making in complex systems.
In Hollnagel (editor), *Intelligent Decision Support in Process Environments*. Springer Verlag, New York, N.Y., 1986.

[8] Hackman, J.R. and Kaplan, R.E.
Interventions into Group Process: An Approach to Improving the Effectiveness of Groups.
Decision Science 5:459-480, 1974.

[9] Hirokawa, R.Y., and Pace, R.A.
A DEscriptive Investigation of the possible Communication-based Reasons for Effective and Ineffective Group Decision Making.
Communication Monographs 50:363-379, 1983.

[10] Huhns, M., Bridgeland, D.
Distributed Truth Maintenance.
In *Proceedings of the 10th International Workshop on DAI.* Bandera, Texas, 1990.

[11] Lander, S., and Lesser, V.
A Framework for Cooperative Problem-Solving Among Knowledge-Based Systems.
In *Proceedings of the MIT-JSME Workshop on Cooperative Product Development.* Cambridge, Mass., 1989.

[12] Lesser, V.
A Retrospective View of FA/C Distributed Problem Solving.
IEEE Transactions on System, Man and Cybernetics 21(6):1347-1362, 1991.

[13] Mason, C. and Johnson, R.
DATMS: A Framework for Distributed Assumption Based Reasoning.
In M. Huhns and L. Gasser (editor), *Distributed Artificial Intelligence, Volume II.* Pittman Publishing Ltd and Morgan Kaufmann, 1989.

[14] Miao, X., Luh, P.B., Kleinman, D.L.
A Normative-Descriptive Approach to Hierarchical Team Resource Allocation.
IEEE Transactions on System, Man and Cybernetics 22(3):482-497, 1992.

[15] Orasanu, J.
Shared Mental Models and Crew Performance.
In *Proceedings of the 34 Annual Meeting of the Human Factors Society.* Orlando, Fla., 1990.

[16] Sycara, K.
Multi-Agent Compromise via Negotiation.
In M. Huhns and L. Gasser (editor), *Distributed Artificial Intelligence, Volume II.* Pittman Publishing Ltd and Morgan Kaufmann, 1989.

[17] Sycara, K.
Problem Restructuring in Negotiation.
Management Science 37(10):1248-1268, 1991.

[18] Sycara, K., and Lewis, C.M.
Modeling Group Decision Making and Negotiation in Concurrent Product Design.
Systems Automation: Research and Applications 1(3):217-238, 1991.

[19] Sycara, K. and Lewis, M.
Forming Shared Mental Models.
In *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society,* pages 400-405. Chicago, Ill., 1991.

[20] Werkman, K.
Knowledge Based Model of Negotiation Using Shareable Perspectives.
In *Proceedings of the 10th International Workshop on DAI.* Bandera, Texas, 1990.

DISTRIBUTION LIST

addresses	number of copies
MR. ALBERT G. FRANTZ RL/C3CA BLDG #3 525 BROOKS ROAD GRIFFISS AFB NY 13441-4505	10
MR. STEVE SMITH CARNEGIE MELLON UNIVERSITY SCHOOL OF COMPUTER SCIENCE PITTSBURGH PA 15213	5
RL/SUL TECHNICAL LIBRARY 26 ELECTRONIC PKY GRIFFISS AFB NY 13441-4514	1
ADMINISTRATOR DEFENSE TECHNICAL INFO CENTER DTIC-FDAC CAMERON STATION BUILDING 5 ALEXANDRIA VA 22304-6145	2
ADVANCED RESEARCH PROJECTS AGENCY 3701 NORTH FAIRFAX DRIVE ARLINGTON VA 22203-1714	1
NAVAL WARFARE ASSESSMENT CENTER GIDEP OPERATIONS CENTER/CODE QA-50 ATTN: E RICHARDS CORONA CA 91718-5000	1
HQ ACC/DRIY ATTN: MAJ. DIVINE LANGLEY AFB VA 23665-5575	1
WRIGHT LABORATORY/AAAI-4 WRIGHT-PATTERSON AFB OH 45433-6543	1

WRIGHT LABORATORY/AAAI-2 1
ATTN: MR FRANKLIN HUTSON
WRIGHT-PATTERSON AFB OH 45433-6543

AFIT/LDEE 1
2950 P STREET
WRIGHT-PATTERSON AFB OH 45433-6577

WRIGHT LABORATORY/MTEL 1
WRIGHT-PATTERSON AFB OH 45433

AAMRL/HE 1
WRIGHT-PATTERSON AFB OH 45433-6573

AUL/LSE 1
BLDG 1405
MAXWELL AFB AL 36112-5564

US ARMY STRATEGIC DEF 1
CSSD-IM-PA
PO BOX 1500
HUNTSVILLE AL 35807-3801

COMMANDING OFFICER 1
NAVAL AVIONICS CENTER
LIBRARY D/765
INDIANAPOLIS IN 46219-2189

COMMANDING OFFICER 1
NCCOSC RDTE DIVISION
CODE 0274B, TECH LIBRARY
53560 HULL STREET
SAN DIEGO CA 92152-5001

CMDR 1
NAVAL WEAPONS CENTER
TECHNICAL LIBRARY/C3431
CHINA LAKE CA 93555-6001

SPACE & NAVAL WARFARE SYSTEMS COMM 1
WASHINGTON DC 20363-5100

CDR, U.S. ARMY MISSILE COMMAND 2
REDSTONE SCIENTIFIC INFO CENTER
AMSMI-RD-CS-R/ILL DOCUMENTS
REDSTONE ARSENAL AL 35898-5241

ADVISORY GROUP ON ELECTRON DEVICES 2
ATTN: DOCUMENTS
2011 CRYSTAL DRIVE, SUITE 307
ARLINGTON VA 22202

LOS ALAMOS NATIONAL LABORATORY 1
REPORT LIBRARY
MS 5000
LOS ALAMOS NM 87544

AEDC LIBRARY 1
TECH FILES/MS-100
ARNOLD AFB TN 37389

COMMANDER/USAISC 1
ATTN: ASOP-DO-TL
BLDG 61801
FT HUACHUCA AZ 85613-5000

AIR WEATHER SERVICE TECHNICAL LIB 1
FL 4414
SCOTT AFB IL 62225-5458

AFIWC/MSO 1
102 HALL BLVD STE 315
SAN ANTONIO TX 78243-7016

SOFTWARE ENGINEERING INST (SEI) 1
TECHNICAL LIBRARY
5000 FORBES AVE
PITTSBURGH PA 15213

DIRECTOR NSA/CSS 1
W157
9800 SAVAGE ROAD
FORT MEADE MD 21055-6000

NSA 1
E323/HC
SAB2 DOOR 22
FORT MEADE MD 21055-6000

NSA 1
ATTN: D. ALLEY
DIV X911
9800 SAVAGE ROAD
FT MEADE MD 20755-6000

DOD 1
R31
9800 SAVAGE ROAD
FT. MEADE MD 20755-6000

DIRNSA 1
R509
9800 SAVAGE ROAD
FT MEADE MD 20775

ESC/IC 1
50 GRIFFISS STREET
HANSCOM AFB MA 01731-1619

ESC/AV 1
20 SCHILLING CIRCLE
HANSCOM AFB MA 01731-2816

FL 2807/RESEARCH LIBRARY 1
OL AA/SULL
HANSCOM AFB MA 01731-5000

TECHNICAL REPORTS CENTER 1
MAIL DROP 0130
BURLINGTON ROAD
BEDFORD MA 01731

DEFENSE TECHNOLOGY SEC ADMIN (DTSA) 1
ATTN: STTD/PATRICK SULLIVAN
400 ARMY NAVY DRIVE
SUITE 300
ARLINGTON VA 22202

MR. GLENN ABRETT 1
10 ROCKTOWN RD
LAMBERTVILLE NJ 08530

MS. KAREN ALGUIRE 1
RL/C3CA
525 BROOKS RD
GRIFFISS AFB NY 13441-4505

JAMES ALLEN 1
COMPUTER SCIENCE DEPT/BLDG RM 709
UNIV OF ROCHESTER
WILSON BLVD
ROCHESTER NY 14627

MR. ROGER ALEX 1
DIGITAL SYSTEMS RSCH INC
4301 NORTH FAIRFAX DRIVE
SUITE 725
ARLINGTON VA 22203

YIGAL ARENS 1
USC-ISI
4676 ADMIRALTY WAY
MARINA DEL RAY CA 90292

RICK ARTHUR 1
GE CORP, R&D
K1-5C29B
P.O. BOX 8
SCHENECTADY NY 12301

MR. RAY BAREISS 1
THE INST. FOR LEARNING SCIENCES
NORTHWESTERN UNIV
1890 MAPLE AVE
EVANSTON IL 60201

MR. JEFF BERLINER BBN SYSTEMS & TECHNOLOGIES 10 MOULTON STREET CAMBRIDGE MA 02138	1
MARIE A. BIENKOWSKI SRI INTERNATIONAL 333 RAVENSWOOD AVE/EK 337 MENLO PRK CA 94025	1
MR. MARK S. BODDY HONEYWELL SYSTEMS & RSCH CENTER 3660 TECHNOLOGY DRIVE MINNEAPOLIS MN 55418	1
PIERO P. BONISSONE GE CORPORATE RESEARCH & DEVELOPMENT BLDG K1-RM 5C-32A P. O. BOX 8 SCHENECTADY NY 12301	1
MR. JOHN BREESE ROCKWELL INTERNATIONAL SCIENCE CTR 444 HIGH STREET PALO ALTO CA 94301	1
MR. GLENN BROUSSARD 12914B GRAYS POINTE FAIRFAX VA 22033	1
MR. DAVID BROWN MITRE EAGLE CENTER 3, SUITE 8 O'FALLON IL 62269	1
MR. MARK BURSTEIN BBN SYSTEMS & TECHNOLOGIES 10 MOULTON STREET CAMBRIDGE MA 02138	1
MR. GREGG COLLINS INST FOR LEARNING SCIENCES 1890 MAPLE AVE EVANSTON IL 60201	1

MR. RANDALL J. CALISTRI-YEH 1
ORA CORPORATION
301 DATES DRIVE
ITHACA NY 14850-1313

LT COL STEPHEN E. CROSS 1
ARPA/SISTO
3701 N. FAIRFAX DR., 7TH FLOOR
ARLINGTON VA 22209-1714

MR. JIM CARCIOFINI 1
HONEYWELL SYSTEMS & RESEARCH
CENTER MN65-2100
3660 TECHNOLOGY DRIVE
MINNEAPOLIS MN 55418

MS. JUDITH DALY 1
ARPA/ASTO
3701 N. FAIRFAX DR., 7TH FLOOR
ARLINGTON VA 22209-1714

THOMAS CHEATHAM 1
HARVARD UNIVERSITY
DIV OF APPLIED SCIENCE
AIKEN, RM 104
CAMBRIDGE MA 02138

MS. LAURA DAVIS 1
CODE 5510
NAVY CTR FOR APPLIED RES IN AI
NAVAL RESEARCH LABORATORY
WASH DC 20375-5337

MS. GLADYS CHOW 1
COMPUTER SCIENCE DEPT.
UNIV OF CALIFORNIA
LOS ANGELES CA 90024

THOMAS L. DEAN 1
BROWN UNIVERSITY
DEPT OF COMPUTER SCIENCE
P.O. BOX 1910
PROVIDENCE RI 02912

WESLEY CHU 1
COMPUTER SCIENCE DEPT
UNIV OF CALIFORNIA
LOS ANGELES CA 90024

MR. ROBERTO DESIMONE 1
SRI INTERNATIONAL (EK335)
333 RAVENSWOOD AVE
MENLO PRK CA 94025

PAUL R. COHEN 1
UNIV OF MASSACHUSETTS
COINS DEPT
LEDERLE GRC
AMHERST MA 01003

MS. MARIE DEJARDINS 1
SRI INTERNATIONAL
333 RAVENSWOOD AVENUE
MENLO PRK CA 94025

JON DOYLE 1
LABORATORY FOR COMPUTER SCIENCE
MASS INSTITUTE OF TECHNOLOGY
545 TECHNOLOGY SQUARE
CAMBRIDGE MA 02139

DR. ROBERT A. FLEMING 1
NAVAL CMD, CONTROL & OCEAN SURV CTR
R&D TEST & EVALUATION DIVISION
53570 SILVERGATE AVE., RM 1405
SAN DIEGO CA 92152-5146

DR. BRIAN DRABBLE 1
AI APPLICATIONS INSTITUTE
UNIV OF EDINBURGH/80 S. BRIDGE
EDINBURGH EH1 LHN
UNITED KINGDOM

MR. SCOTT FOUSE 1
ISX CORPORATION
4353 PARK TERRACE DRIVE
WESTLAKE VILLAGE CA 91361

MR. STU DRAPER 1
MITRE
EAGLE CENTER 3, SUITE 8
O'FALLON IL 62269

NORTHRUP FOWLER III 1
RL/C3C
525 BROOKS ROAD
GRIFFISS AFB NY 13441-4505

MARK FOX 1
DEPT O INDUSTRIAL ENGRG
UNIV OF TORONTO
4 TADDLE CREAK ROAD
TORONTO, ONTARIA, CANADA

MR. GARY EDWARDS 1
4353 PARK TERRACE DRIVE
WESTLAKE VILLACA 91361

MR. ALBERT G. FRANTZ 1
RL/C3CA
525 BROOKS RD
GRIFFISS AFB NY 13441-4505

MS. MARTHA FARINACCI 1
MITRE
7525 COLSHIRE DRIVE
MCLEAN VA 22101

MR. RUSS FREW 1
GENERAL ELECTRIC
MOORESTOWN CORPORATE CENTER
BLDG ATK 145-2
MOORESTOWN NJ 08057

MICHAEL FEHLING 1
STANFORD UNIVERSITY
ENGINEERING ECO SYSTEMS
STANFORD CA 94305

MR. RICH FRITZSON 1
CENTER OR ADVANCED INFO TECHNOLOGY
UNISYS
P.O. BOX 517
PAOLI PA 19301

MS. KRISTIAN J. HAMMOND 1
UNIV OF CHICAGO
COMPUTER SCIENCE DEPT/RY155
1100 E. 58TH STREET
CHICAGO IL 60637

MR. ROBERT FROST 1
MITRE CORP
WASHINGTON C3 CENTER, MS 644
7525 COLSHIER ROAD
MCLEAN VA 22101-3481

RICK HAYES-ROTH CIMFLEX-TEKNOWLEDGE 1810 EMBARCADERO RD PALO ALTO CA 94303	1
RANDY GARRETT INST FOR DEFENSE ANALYSES (IDA) 1801 N. BEAUREGARD STREET ALEXANDRA VA 22311-1772	1
MR. JIM HENDLER UNIV OF MARYLAND DEPT OF COMPUTER SCIENCE OLLEGE PARK MD 20742	1
MS. YOLANDA GIL USC/ISI 4676 ADMIRALTY WAY MARINA DEL RAY CA 90292	1
MR.MAX HERION ROCKWELL INTERNATIONAL SCIENCE CTR 444 HIGH STREET PALO ALTO CA 94301	1
MATHEW L. GINSBERG STANFORD UNIVERSITY COMPUTER SCIENCE DEPARTMENT STANFORD CA 94305	1
MR.JEFF HERMAN NAVAL CMD, CONTROL & OCEAN SURV CTR R&D, TEST & EVAL CENTER CODE 444\SAN DIEGO CA 92152-5000	1
MR. STEVE GOYA DISA/JIEO/GS11 CODE TBD 11440 ISAAC NEWTON SQ RESTON VA 22090	1
MR. MARTON A. HIRSCHBERG 207 BRIAR CLIFF LANE BEL AIR MD 21014	1

MS. LAUREN HALVERSON GENERAL ELECTIC CRD P. O. BOX 8, KL-5C31B SCHENECTADY NY 12301	1
MR. MORTON A. HIRSCHBERG, DIRECTOR US ARMY RESEARCH LABORATORY ATTN: AMSRL-CI-CB ABERDEEN PROVING GROUND MD 21005-5066	1
MR. GREG KONRAD USOCOM/SOJ5-PJ MACDILL AFB FL 33608	1
MR. MARK A. HOFFMAN ISX CORPORATION 1165 NORTHCHASE PARKWAY MARIETTA GA 30067	1
MR. RON LARSEN NAVAL CMD, CONTROL & OCEAN SUR CTR RESEARCH, DEVELOP, TEST & EVAL DIV CODE 444 SAN DIEGO CA 92152-5000	1
DR. JAMES JUST MITRE DEPT. W032-M/S Z360 7525 COLSHIER RD MCLEAN VA 22101	1
PETER KARP SRI INTERNATIONAL ARTIFICIAL INTELLIGENCE CENTEER 333 RAVENSWOOD AVE MENLO PARK CA 94025	1
MS. NANCY B. LEHRER ISX CORPORATION 4353 PARK TERRACE DRIVE WESTLAKE VILLAGE CA 91361	1
LTC THOMAS E. KAZMIERCZAK TCJ5-OP JOPES PROJ GRP BLDG 1961 SCOTT AFB IL 62225-5001	1

MR. TED LINDEN ADVANCED DECISION SYSTEMS 1500 PLYMOUTH STREET MOUNTAIN VIEW A 94043-1230	1
MR. CRAIG KNOBLOCK USC-ISI 4676 ADMIRALTY WAY MARINA DEL RAY CA 90292	1
MR. RICHARD LOWE (AP-10) SRA CORPORATION 2000 15TH STREET NORTH ARLINGTON VA 22201	1
MR. TED C. KRAL BBN SYSTEMS & TECHNOLOGIES 4015 HANCOCK STREET, SUITEE 101 SAN DIEGO CA 92110	1
MR. JOHN LOWRENCE SRI INTERNATIONAL ARTIFICIAL INTELLIGENCE CENTER 333 RAVENSWOOD AVE MENLO PARK CA 94025	1
DR. ALAN MEYROWITZ NAVAL RESEARCH LABORATORY/CODE 5510 4555 OVERLOOK AVE WASH DC 20375	1
ALICE MULVEHILL MITRE CORPORATION BURLINGTON RD M/S K-302 BEDFORD MA 01730	1
ROBERT MACGREGOR USC/ISI 4676 ADMIRALTY WAY MARINA DEL REY CA 90292	1
WILLIAM S. MARK, MGR AI CENTER LOCKHEED MISSILES & SPACE CENTER 1801 PAGE MILL RD PALO ALTO CA 94304-1211	1

RICHARD MARTIN 1
SOTWARE ENGINEERING INSTITUTE
CARNEGIE MELLON UNIV
PITTSBURGH PA 16213

MR. DAVE NOBLE 1
NAVAL CMD, CONTROL & OCEAN SURV CTR
ENGRG RESEARCH/CODE 444
R&D, TEST & EVAL CENTER
SAN DIEGO CA 92152-5000

MR. MICHAEL MATTOCK 1
THE RAND CORP
1700 MAIN STREET
SANTA MONICA CA 90407-2138

MR. CHRISTOPHER OWENS 1
UNIV OF CHICAGO
DEPT OF COMPUTER SCIENCE
110 E. 58TH STREET
CHICAGO IL 60637

DRW MCDERMOTT 1
YALE COMPUTER SCIENCE DEPT
P.O. BOX 2158, YALE STATION
51 PROPSPECT STREET
NEW HAVEN CT 06520

MS. CECILE PARIS 1
USC/ISI
4676 ADMIRALTY WAY
MARINA DEL RAY CA 90292

DOUGLAS SMITH 1
KESTREL INSTITUTE
3260 HILLVIEW AVE
PALO ALTO CA 94304

DR. AUSTIN TATE 1
AI APPLICATIONS INSTITUTE
UNIV OF EDINBURGH
80 SOUTH BRIDGE
EDINBURGH EH1 1HN - SCOTLAND

MS. REGINA SMITH 1
MERIDIAN CORPORATION
4001 NORTH FAIRFAX DRIVE
SUITE 200
ARLINGTON VA 2203-1714

EDWARD THOMPSON ARPA/SISTO 3701 N. FAIRFAX DR., 7TH FL ARLINGTON VA 22209-1714	1
MR. STEPHEN F. SMITH ROBOTICS INSTITUTE/CMU SCHENLEY PRK PITTSBURGH PA 15213	1
MR. MNUELA VELOSO CMU SCHOOL O COMPUTER SCIENCE PITTSBURGH PA 15213-3891	
LTCOL RAYMOND STACHA DEPUTY SCIENTIFIC & TECHNICAL ADVISOR HQ USCINCPAC/STA CAMP H. M. SMITH HI 96861	1
DR. ABRAHAM WAKSMAN AFOSR/NM 110 DUNCAN AVE., SUITE B115 BOLLING AFB DC 20331-0001	1
JONATHAN P.STILLMAN GENERAL ELECTRIC CRD 1 RIVER RD, RM K1-5C31A P. O. BOX 8 SCHENECTADY NY 12345	1
MR. EDWARD C. T. WALKER BBN SYSTEMS & TECHNOLOGIES 10 MOULTON STREET CAMBRIDGE MA 02138	1
MR. BILL SWARTOUT USC/ISI 4676 ADMIRALTY WAY MRINA DEL RAY CA 90292	1
GIO WIEDERHOLD PROGRAM MGR FOR KB SYSTEMS ARPA/SISTO 3701 NORTH AIRFX DRIVE, RM 739 ARLINGTON VA 22203-1714	1

KATIA SYCARA/THE ROBOTICS INST 1
SCHOOL OF COMPUTER SCIENCE
CARNEGIE MELLON UNIV
DOHERTY HALL RM 3325
PITTSBURGH PA 15213

MR. DAVID E. WILKINS 1
SRI INTERNATIONAL
ARTIFICIAL INTELLIGENCE CENTER
333 RAVENSWOOD AVE
MENLO PRK CA 94025

DR. PATRICK WINSTON 1
MASS INSTITUTE OF TECHNOLOGY
RM NE43-817
545 TECHNOLOGY SQUARE
CAMBRIDGE MA 02139

HUA YANG 1
COMPUTER SCIENCE DEPT
UNIV OF CALIORNIA
LOS ANGELES CA 90024

LTCOL DAVE NEYLAND 1
ARPA/ISTO
3701 N. FAIRFAX DRIVE, 7TH FLOOR
ARLINGTON VA 22209-1714

MR. RICK SCHANTZ 1
BBN SYSTEMS & TECHNOLOGIES
10 MOULTON STREET
CAMBRIDGE MA 02138

LTC FRED M. RAWCLIFFE 1
USTRANSCOM/TCJ5-SC
BLDG 1900
SCOTT AFB IL 62225-7001

JOHN P. SCHILL 1
NAVAL COMMAND, CONTROL & OCEAN
SURVEILLANCE CENTER/CODE 423
EVALUATION DIVISION
SAN DIEGO CA 92152-5000

MR. DONALD F. ROBERTS 1
RL/C3CA
BLDG 3
525 BROOKS RD
GRIFFISS AFB NY 13441-4505

MR. DAVE SCHNEEGAS 1
USPACOM/J3
CAMP SMITH, HI 9686L-5025

ALLEN SEARS 1
MITRE
7525 COLESHIRE DRIVE, STOP Z289
MCLEAN VA 22101

STEVE ROTH 1
CENTER FOR INTEGRATED MANUFACTURING
THE ROBOTICS INSTITUTE
CARNEGIE MELLON UNIV
PITTSBURGHJ PA 15213-3890

JEFF ROTHENBERG 1
SENIOR COMPUTER SCIENTIST
THE RAND CORPORATION
1700 MAIN STREET
SANTA MONICA CA 90407-2138

YOAV SHOHAM 1
STANFORD UNIVERSITY
COMPUTER SCIENCE DEPT
STANFORD CA 94305

MR. DAVID B. SKALAK 1
UNIV OF MASSACHUSETTS
DEPT OF COMPUTER SCIENCE
RM 243, LGRC
AMHERST MA 01003

MR. BILL ROUSE 1
SEARCH TECHNOLOGY
4725 PEACH TREE CORNER CIRCLE
SUITE 200
NORCROSS GA 30092

MR. MIKE ROUSE 1
AFSC
7800 HAMPTON RD
NORFOLD VA 23511-6097

MR. DAVID E. SMITH 1
ROCKWELL INTERNATIONAL
444 HIGH STREET
PALO ALTO CA 94301

JEFF ROTHENBERG
SENIOR COMPUTER SCIENTIST
THE RAND CORPORATION
1700 MIN STREET
SANTA MONIA CA 90407-2138

1